

SeeBeyond ICAN Suite

e*Way Intelligent Adapter for Siebel EIM User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

Java Version



The information contained in this document is subject to change and is updated periodically to reflect changes to the applicable software. Although every effort has been made to ensure the accuracy of this document, SeeBeyond Technology Corporation (SeeBeyond) assumes no responsibility for any errors that may appear herein. The software described in this document is furnished under a License Agreement and may be used or copied only in accordance with the terms of such License Agreement. Printing, copying, or reproducing this document in any fashion is prohibited except in accordance with the License Agreement. The contents of this document are designated as being confidential and proprietary; are considered to be trade secrets of SeeBeyond; and may be used only in accordance with the License Agreement, as protected and enforceable by law. SeeBeyond assumes no responsibility for the use or reliability of its software on platforms that are not supported by SeeBeyond.

SeeBeyond, e*Gate, e*Way, and e*Xchange are the registered trademarks of SeeBeyond Technology Corporation in the United States and/or select foreign countries. The SeeBeyond logo, SeeBeyond Integrated Composite Application Network Suite, eGate, eWay, eInsight, eVision, eXchange, eView, eIndex, eTL, ePortal, eBAM, and e*Insight are trademarks of SeeBeyond Technology Corporation. The absence of a trademark from this list does not constitute a waiver of SeeBeyond Technology Corporation's intellectual property rights concerning that trademark. This document may contain references to other company, brand, and product names. These company, brand, and product names are used herein for identification purposes only and may be the trademarks of their respective owners.

© 2005 SeeBeyond Technology Corporation. All Rights Reserved. This work is protected as an unpublished work under the copyright laws.

This work is confidential and proprietary information of SeeBeyond and must be maintained in strict confidence.

Version 20050406041317.

Contents

Preface	9
Intended Reader	9
Organization	9
Nomenclature	10
Online Use	10
Writing Conventions	10
Additional Documentation	11
<hr/>	
Chapter 1	
Introduction	12
Overview	12
Communicating with Siebel	13
Siebel EIM	13
IFB Control File	13
e*Way Operation	14
Siebel to e*Gate	14
e*Gate to Siebel	15
e*Way Components	16
Supported Operating Systems	17
<hr/>	
Chapter 2	
Installation	18
System Requirements	18
External System Requirements	19
Siebel	19
RDBMS	19
External Configuration Requirements	19
Installing the e*Way	20

Windows Systems	20
Installation Procedure	20
Subdirectories and Files	22
Environment Configuration	22
UNIX Systems	23
Installation Procedure	23
Subdirectories and Files	24
Environment Configuration	24
Optional Example Files	25
Installation Procedure	25
Subdirectories and Files	26
Siebel 2000	26
Siebel 7	29

Chapter 3

System Implementation	33
Overview	33
Pre-Implementation Tasks	33
Known Cautions	33
Implementation Sequence	34
Viewing e*Gate Components	34
Creating a Schema	35
Creating Event Types	36
Generating Event Type Definitions	37
Using the DBWizard ETD Builder	37
The Generated ETDs	45
Editing an Existing ETD Using the Database Wizard	45
Using the Timestamp Node	46
Assigning ETDs to Event Types	47
Creating Intelligent Queues	48
Defining Collaborations	49
The Java Collaboration Rules Editor	49
Subcollaboration Rules	49
Auditing	50
Error logging—Inbound to Siebel	50
Error logging—Outbound from Siebel	50
Journaling	50
Sample Schema	51
EIM_Extract	52
Architecture	52
Operation	52
EIM_Post	53
Architecture	53
Operation	53

Chapter 4

e*Way Setup	54
Overview	54
Setting Up the e*Way	55
Creating the e*Way	55
Modifying e*Way Properties	56
Configuring the e*Way	57
Changing the User Name	58
Setting Startup Options or Schedules	58
Activating or Modifying Logging Options	60
Activating or Modifying Monitoring Thresholds	61
Creating e*Way Connections	62
Managing e*Way Connections	66
Controlling When a Connection is Made	66
Controlling When a Connection is Disconnected	67
Controlling the Connectivity Status	67
Using the e*Way Configuration Editor	68
Section and Parameter Controls	69
Parameter Configuration Controls	69
File Menu Selections	70
Command-line Configuration	70
Getting Help	70
Troubleshooting the e*Way	71
Configuration Problems	71
System-related Problems	72

Chapter 5

Operational Overview	73
Siebel EIM	73
IFB Configuration File	75
Siebel EIM e*Way Operation	76
Siebel to e*Gate	76
Overview	76
Export Procedure	77
Example	77
e*Gate to Siebel	81
Overview	81
Import Procedure	82
Example	83
Event Type Definitions	89
SiebelEIM	89
IFB_Control	89
IFB_Content	91

IFB_Iteration	92
EIM_Control	93
Basic e*Gate/e*Way Operation	96
Multi-Mode e*Way	96
Event Type Definitions	98
ETD Components	98
ETD Builders	98
Collaborations	100
Java Collaboration Service	101
e*Way Connections	102
Establishing Connections	102

Chapter 6

Configuration Parameters 104

Overview	104
Multi-Mode e*Way	104
e*Way Connections	104
Siebel EIM	104
JDBC Oracle	105
DB2 Universal Database	105
Microsoft SQL Server	105
Multi-Mode e*Way	106
JVM Settings	106
JNI DLL Absolute Pathname	106
CLASSPATH Prepend	106
CLASSPATH Override	107
CLASSPATH Append From Environment Variable	107
Initial Heap Size	108
Maximum Heap Size	108
Maximum Stack Size for Native Threads	108
Maximum Stack Size for JVM Threads	108
Class Garbage Collection	109
Garbage Collection Activity Reporting	109
Asynchronous Garbage Collection	109
Report JVM Info and all Class Loads	109
Disable JIT	109
Remote debugging port number	110
Suspend option for debugging	110
General Settings	111
Rollback Wait Interval	111
Siebel EIM e*Way Connections	112
Connector	112
type	112
class	112
Siebel Server	113
Siebel Version	113
Database Client Connectivity	113
Database Host Name	113
Database Table Owner	113
Database User Name	114
Database Password	114
Gateway Server	114
Enterprise Server Name	114

Administrator User Name	114
Administrator Password	115
Siebel Server Name	115
Path to Siebel Server	115
Siebel Control	116
Batch Number	116
IFB Filename	116
Siebel System Import	117
Siebel State File	117
Siebel System Export	118
Last Time of Export File	118
Siebel State File	118
JDBC Oracle e*Way Connections	119
DataSource Settings	119
class	119
DriverType	119
ServerName	120
PortNumber	120
DatabaseName	120
user name	120
password	120
timeout	121
Connector Settings	122
type	122
class	122
transaction mode	122
connection establishment mode	122
connection inactivity timeout	123
connection verification interval	123
DB2 UDB e*Way Connections	124
DataSource Settings	124
class	124
Connection Method	124
CollectionID	124
DatabaseName	125
LocationName	125
PackageName	125
user name	125
password	125
PortNumber	126
ServerName	126
timeout	126
data source attribute value pair separator	126
data source attributes	127
Connector Settings	128
type	128
class	128
transaction mode	128
connection establishment mode	128
connection inactivity timeout	129
connection verification interval	129
SQL Server e*Way Connections	130
DataSource Settings	130
class	130
Connection Method	130
ServerName	130
PortNumber	131
DatabaseName	131
user name	131
password	131

SelectMethod	131
timeout	132
Connector Settings	133
type	133
class	133
transaction mode	133
connection establishment mode	133
connection inactivity timeout	134
connection verification interval	134

Chapter 7

Java Classes and Methods **135**

Overview **135**

Object Classes **136**

EIM_Control Class **137**

Methods **137**

getState	137
getStatus	138
startTask	138
startTaskAndWaitUntilDone	139
waitUntilDone	139

IFB_Control Class **141**

Methods **141**

generateIb	141
getImportState	141
setImportState	142
getExportState	142
setExportState	143
getLastExportTime	143
setLastExportTime	144

Index **145**

Preface

This Preface contains information regarding the User's Guide itself.

P.1 Intended Reader

The reader of this guide is presumed to be a developer or system administrator with responsibility for maintaining the SeeBeyond™ e*Gate™ Integrator system, and have a working knowledge of:

- Operation and administration of the appropriate operating systems (see [Supported Operating Systems](#) on page 17)
- Windows-style GUI operations
- Siebel EIM concepts and operations
- Integrating Siebel EIM applications with external systems

P.2 Organization

This User's Guide is organized into two parts. The first part, consisting of Chapters 1-5, introduces the e*Way and describes the procedures for installing and setting up the e*Way, and implementing a working system incorporating the e*Way. This part should be of particular interest to a System Administrator or other user charged with the task of getting the system up and running.

The second part, consisting of Chapters 6-8, describes the details of e*Way operation and configuration, including descriptions of the exposed Java methods. This part should be of particular interest to a Developer involved in customizing the e*Way for a specific purpose. Information contained in this part that is necessary for the initial setup of the e*Way is cross-referenced in the first part of the guide, at the appropriate points in the procedures.

P.3 Nomenclature

Note that for purposes of brevity, the e*Way Intelligent Adapter for Siebel EIM is frequently referred to as the Siebel EIM e*Way, or simply the e*Way.

P.4 Online Use

This User's Guide is provided in Adobe Acrobat's Portable Document Format (PDF). As such, it can be printed out on any printer or viewed online. When viewing online, you can take advantage of the extensive hyperlinking imbedded in the document to navigate quickly throughout the Guide.

Hyperlinking is available in:

- The Table of Contents
- The Index
- Within the chapter text, indicated by **blue print**

Existence of a hyperlink *hotspot* is indicated when the hand cursor points to the text. Note that the hotspots in the Index are the *page numbers*, not the topics themselves. Returning to the spot you hyperlinked from is accomplished by right-clicking the mouse and selecting **Go To Previous View** on the resulting menu.

P.5 Writing Conventions

The writing conventions listed in this section are observed throughout this document.

Monospaced (Courier) Font

Computer code and text to be typed at the command line are set in Courier as shown below.

```
Configuration for BOB_Promotion
java -jar ValidationBuilder.jar
```

Variables within a command line, or attributes within a method signature, are set in italics as shown below:

```
stcregutl -rh host-name -un user-name -up password -sf
```

Bold Sans-serif Font

- User Input: Click **Apply** to save, or **OK** to save and close.
- File Names and Paths: In the **Open** field, type **D:\setup\setup.exe**.
- Parameter, Function, and Command Names: The default parameter **localhost** is usually used only for testing.

P.6 Additional Documentation

- Many of the procedures included in this User's Guide are described in greater detail in the *e*Gate Integrator User's Guide*
- For more information on the Java Collaboration Service, see the *e*Gate Integrator Collaboration Services Reference*
- For additional information on the Multi-Mode e*Way, see the *Standard e*Way Intelligent Adapter User's Guide*
- For additional information on the Oracle e*Way, see the *e*Way Intelligent Adapter for Oracle User's Guide*
- For additional information on the SQL Server e*Way, see the *e*Way Intelligent Adapter for SQL Server User's Guide*
- For additional information on the DB2 e*Way, see the *e*Way Intelligent Adapter for DB2 Universal Database User's Guide*
- For information on requirements for the Siebel environment, see the *Siebel System Requirements and Supported Platforms* document for the version of Siebel you are using
- For details regarding Siebel EIM, see the *Siebel Administration Guide*

Introduction

This chapter provides a brief introduction to the SeeBeyond Java e*Way Intelligent Adapter for Siebel EIM.

1.1 Overview

The Java e*Way Intelligent Adapter for Siebel EIM provides bidirectional communications between e*Gate Integrator and the Siebel Enterprise Integration Manager (EIM). The e*Way provides an execution environment that oversees Siebel's EIM processes and dynamically creates Siebel IFB files (it also provides the option of generating IFB files manually). It uses the definitions in the IFB file to execute the command to run the Siebel EIM job, and can populate or extract data from the Siebel interface tables using elements of the SeeBeyond JDBC e*Way.

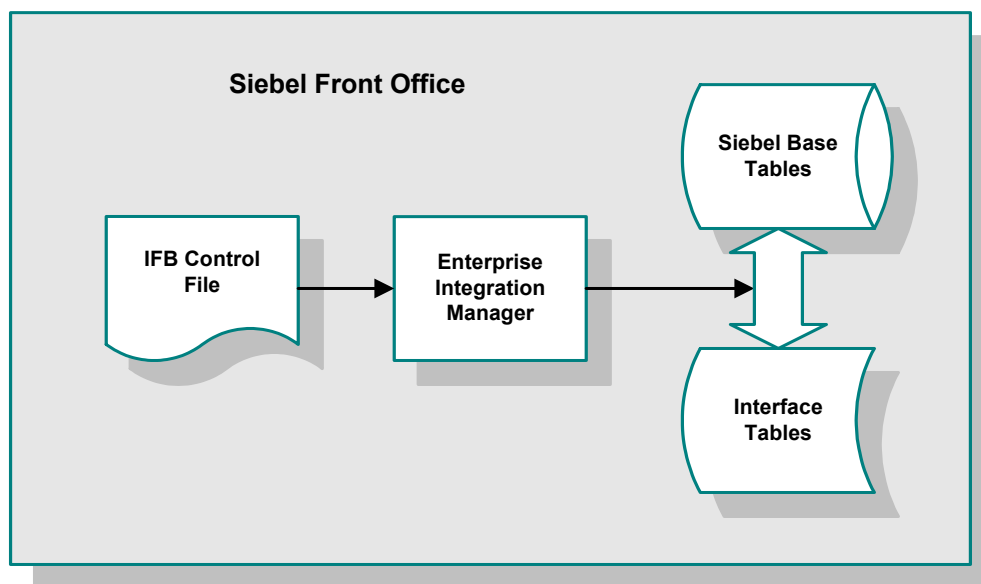
The e*Way also monitors the status of the job—if an error occurs during processing, the error is written into the log file and the e*Way alerts the Schema Manager. It incorporates detailed error logs, bad-Event Journaling, and reprocessing capabilities for the failed imported records during operation of the Siebel Enterprise Integration Manager. All Application Logic and Business Rules are enforced using Siebel EIM.

1.2 Communicating with Siebel

1.2.1 Siebel EIM

The Siebel Enterprise Integration Manager is the first step in exporting, and the last step in importing, data. The database consists of two main groups of tables, interface tables and base tables. The Siebel client application communicates directly with the highly normalized base tables. The interface tables are used as a staging area for importing, exporting, deleting and merging logical groups of data. Each interface table represents a subset of the data in a specific base table.

Figure 1 Siebel Enterprise Integration Manager



When run, EIM coordinates the transfer of data between the base and interface tables. In addition, EIM creates and writes codes and row IDs to the base tables that directly correspond to the complex set of business rules used by the Siebel client application to display data. The EIM process must run on the server also running the Transaction Processor.

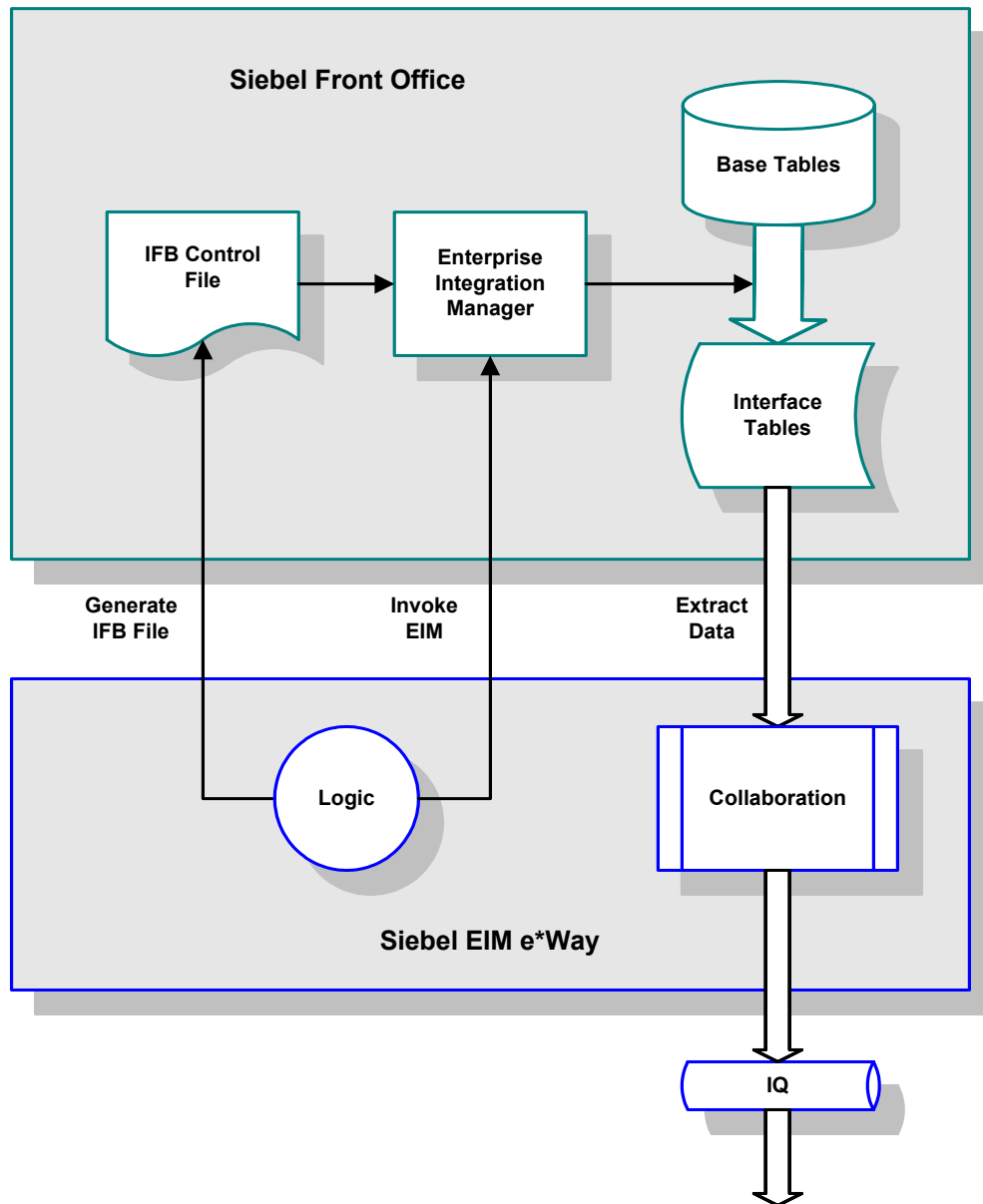
1.2.2 IFB Control File

A control file (*.ifb) is used to determine what data types are loaded and how. The control file follows a certain format—it tells the interface how to log into the database and what process to run. It also lists the columns that the interface does *not* populate (in Siebel-inbound mode), thus preventing erroneous error messages. The IFB control file can be generated automatically by the e*Way or created manually, as configured by the user.

1.3 e*Way Operation

1.3.1 Siebel to e*Gate

Figure 2 Siebel-to-e*Gate Process Flow

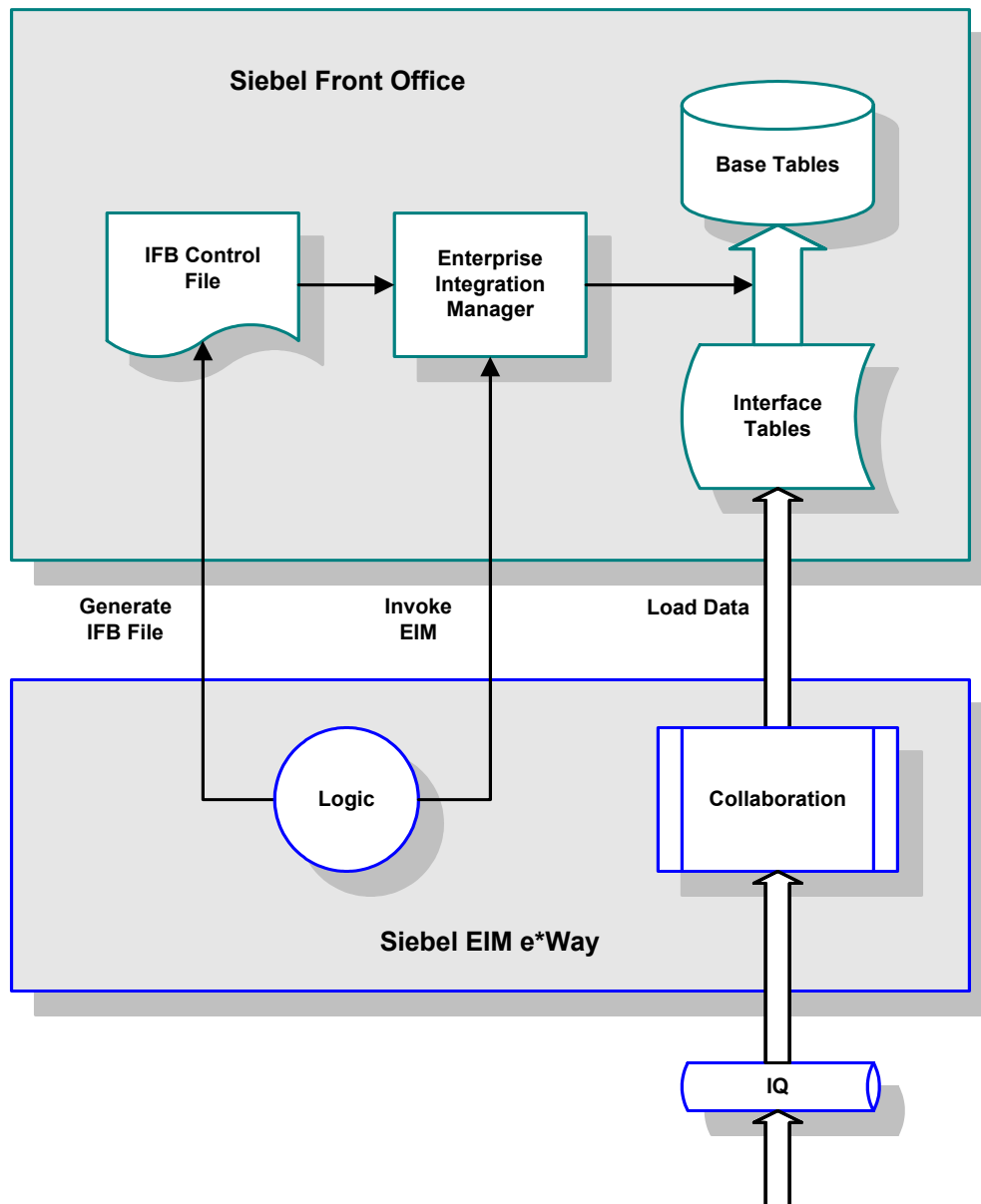


Following a prescribed schedule, the Siebel EIM e*Way sends an IFB file and invokes the Siebel EIM process. Following the definitions in the IFB control file, the EIM copies data from the Siebel Base Tables into the Interface Tables. The e*Way extracts the data from the Interface Tables and maps it into the desired Event Type Definition. The data

then is passed to an Intelligent Queue for subsequent processing and/or routing to the target application by other e*Gate components.

1.3.2 e*Gate to Siebel

Figure 3 e*Gate-to-Siebel Process Flow



Following a prescribed schedule, the Siebel EIM e*Way extracts information from an e*Gate Intelligent Queue. The e*Way generates an IFB control file and sends it to Siebel. It then inserts validated rows of data into the interface table. Records that do not successfully load into the interface table are written to an error file for reprocessing. After the interface tables are populated, the e*Way initiates the Siebel EIM to load the data from the Interface Tables to the Siebel Base Tables.

1.4 e*Way Components

The Java e*Way Intelligent Adapter for Siebel EIM incorporates the following primary components:

- SeeBeyond Multi-Mode e*Way (installed with e*Gate):
 - ♦ stceway.exe
- Default Configuration Definition file:
 - ♦ SiebelEIM.def
- Java classes, contained in the following files:
 - ♦ stcsiebeleim.jar
- Supporting methods and Event Type Definitions

For a list of installed files, see [Installing the e*Way](#) on page 20.

Note: *The e*Gate Schema Designer GUI runs only on the Windows operating system.*

1.5 Supported Operating Systems

The Java e*Way Intelligent Adapter for Siebel EIM currently supports the following combination of operating systems, database management systems, and Siebel versions.

Table 1 English-language Version

Operating System	Siebel Server				Siebel Front Office				
	5.0	6.0	7.0	7.5.3	99.5	99.6	2000	7.0	7.5.3
Windows 2000, Windows XP, and Windows Server 2003	X	X	X	-	X	X	X	X	-
IBM AIX 5.1L and 5.2	-	-	X	-	-	-	-	X	-
Sun Solaris 8	-	-	X	-	-	-	-	X	-

Table 2 Japanese-language Version

Operating System	Siebel		RDBMS		
	2000	7.0	DB2	Oracle	SQL
Windows 2000, Windows XP, and Windows Server 2003	X	X	-	X	X

Table 3 Korean-language Version

Operating System	Siebel		RDBMS		
	2000	7.0	DB2	Oracle	SQL
Windows 2000, Windows XP, and Windows Server 2003	X	X	-	X	X

Installation

This chapter describes the requirements and procedures for installing the e*Way software. Procedures for implementing a working system, incorporating instances of the e*Way, are described in [Chapter 3](#).

Note: Please read the *readme.txt* file located in the *addons\ewsiebeleim* directory on the installation CD-ROM for important information regarding this installation.

2.1 System Requirements

To use the Siebel EIM e*Way, you need the following:

- 1 An e*Gate Participating Host.
- 2 A TCP/IP network connection.
- 3 Sufficient free disk space on both the Participating Host and the Registry Host to accommodate e*Way files (not including sample schemas):
 - ♦ Approximately 7.7 MB on Windows systems
 - ♦ Approximately 6.5 MB on Solaris systems
 - ♦ Approximately 6.7 MB s on AIX systems

Additional disk space is required to process and queue the data that this e*Way processes; the amount necessary varies, based on the type and size of the data being processed.

2.2 External System Requirements

2.2.1 Siebel

To use the e*Way Intelligent Adapter for Siebel EIM, you need the following Siebel software:

- Siebel 2000 or 7.0
 - ♦ Siebel Database Server
 - ♦ Siebel Gateway Server
 - ♦ Siebel Server
- Siebel Tools

Please see the *Siebel System Requirements and Supported Platforms* document for the version of Siebel you are using (see also [Supported Operating Systems](#) on page 17).

2.2.2 RDBMS

To use the e*Way Intelligent Adapter for Siebel EIM, you need one of the following relational database systems installed with Siebel, along with the appropriate client software:

- Oracle 8.1.7
- SQL Server 2000
- DB2 7.2

Please see the *Siebel System Requirements and Supported Platforms* document for the version of Siebel you are using (see also [Supported Operating Systems](#) on page 17).

2.3 External Configuration Requirements

Most installations of Siebel applications require some customization of the Data Model to meet the client's specific requirements. We assume that any customization of Siebel required for your implementation has been performed previous to the installation of e*Gate. No special configuration of the Siebel application is required to interface with e*Gate.

2.4 Installing the e*Way

2.4.1 Windows Systems

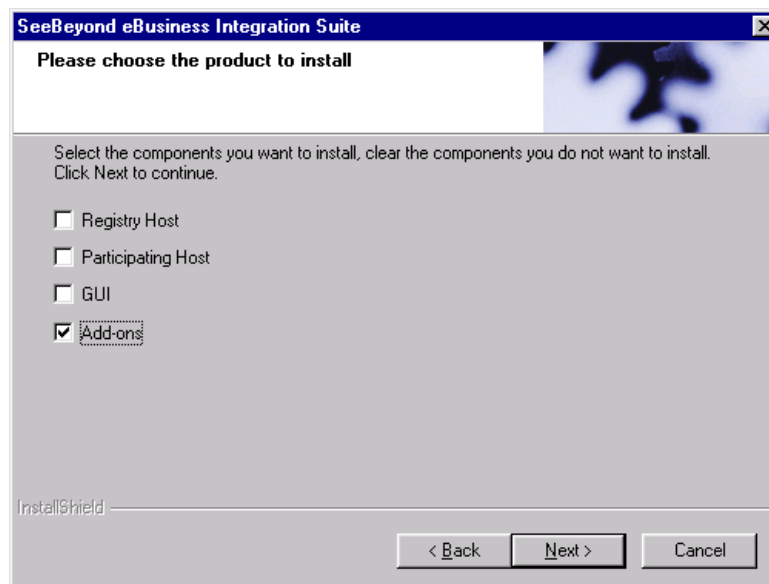
Installation Procedure

Note: *The installation utility detects and suggests the appropriate installation directory. Use this directory unless advised otherwise by SeeBeyond.*

To Install the e*Way on a Microsoft Windows System

- 1 Log in as an Administrator on the workstation on which you want to install the e*Way (you must have Administrator privileges to install this e*Way).
- 2 Exit all Windows programs and disable any anti-virus applications before running the setup program.
- 3 Insert the e*Way installation CD-ROM into the CD-ROM drive.
- 4 Launch the setup program.
 - A If the CD-ROM drive's Autorun feature is enabled, the setup program should launch automatically. Follow the on-screen instructions until the **Choose Product** dialog box appears (see Figure 4). Check **Add-ons**, then click **Next**.

Figure 4 Choose Product Dialog

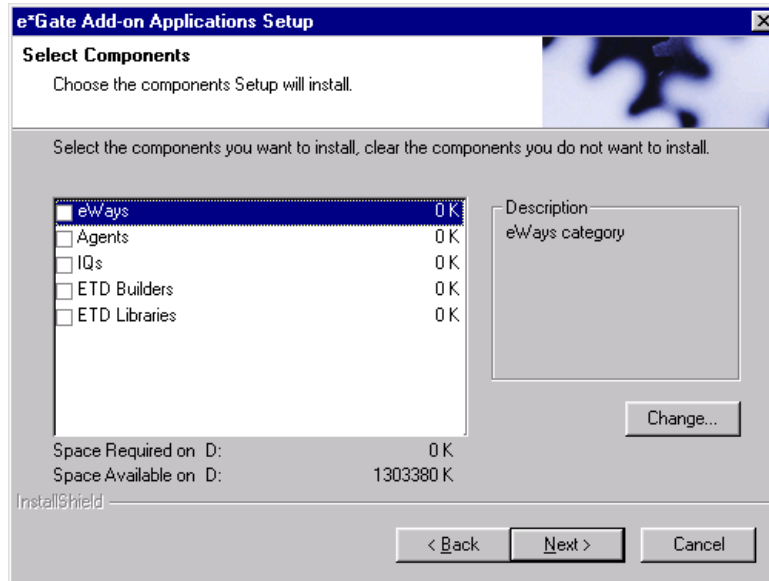


- B If the setup program does not launch automatically, use the Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the following file on the CD-ROM drive (bypassing the **Choose Product** dialog):

setup\addons\setup.exe

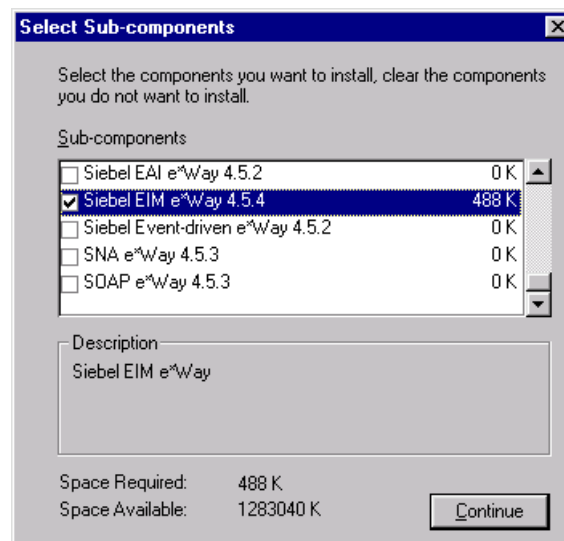
- 5 Follow the on-screen instructions until the **Select Components** dialog box appears (see Figure 5). Highlight—but do not check—**eWays** and then click **Change**.

Figure 5 Select Components Dialog



- 6 When the **Select Sub-components** dialog box appears (see Figure 6), check the **Siebel EIM eWay**.

Figure 6 Select Sub-components Dialog



- 7 Click **Continue**, and the **Select Components** dialog box reappears, showing your selection. Note that several database e*Ways are also listed.
- 8 Click **Next** and continue with the installation.

Subdirectories and Files

Note: *Installing the e*Way Intelligent Adapter for Siebel EIM installs both Java and Monk versions, and also elements of several database e*Ways. Only the files used by the Java version of the Siebel EIM e*Way are listed in this section. The files installed for the other e*Ways are listed in their respective User's Guides.*

By default, the InstallShield installer creates the following subdirectories and installs the following files within the \eGate\client tree on the Participating Host, and the \eGate\Server\registry\repository\default tree on the Registry Host.

Table 4 Participating Host & Registry Host

Subdirectories	Files
\classes\	stcsiebeleim.jar
\configs\siebeleim\	SiebelEIM.def
\etd\	siebeleim.ctl
\etd\siebeleim\	SiebelEIM.xsc

By default, the InstallShield installer also installs the following files within the \eGate\Server\registry\repository\default tree on the Registry Host.

Table 5 Registry Host Only

Subdirectories	Files
\	stcewsiebeleim.ctl

Environment Configuration

No changes are required to the Participating Host's operating environment to support this e*Way.

2.4.2 UNIX Systems

Installation Procedure

Note: *You are not required to have root privileges to install this e*Way. Log on under the user name that you wish to own the e*Way files. Be sure that this user has sufficient privilege to create files in the e*Gate directory tree.*

To Install the e*Way on a UNIX System

- 1 Log onto the workstation containing the CD-ROM drive and, if necessary, mount the drive.
- 2 Insert the e*Way installation CD-ROM into 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 appears. Select the **Install e*Way** option and follow any additional on-screen instructions.

Note: *The installation utility detects and suggests the appropriate installation directory. Use this directory unless advised otherwise by SeeBeyond. Note also that **no spaces** should appear in the installation path name.*

Subdirectories and Files

Note: *Installing the e*Way Intelligent Adapter for Siebel EIM installs both Java and Monk versions, and also elements of several database e*Ways. Only the files used by the Java version of the Siebel EIM e*Way are listed in this section. The files installed for the other e*Ways are listed in their respective User's Guides.*

The preceding installation procedure creates the following subdirectories and installs the following files within the /eGate/client tree on the Participating Host, and the /eGate/Server/registry/repository/default tree on the Registry Host.

Table 6 Participating Host & Registry Host

Subdirectories	Files
/classes/	stcsiebeleim.jar
/configs/siebeleim/	SiebelEIM.def
/etd/	siebeleim.ctl
/etd/siebeleim/	SiebelEIM.xsc

The preceding installation procedure also installs the following files only within the /eGate/Server/registry/repository/default tree on the Registry Host.

Table 7 Registry Host Only

Subdirectories	Files
/	stcewsiebeleim.ctl

Environment Configuration

No changes are required to the Participating Host's operating environment to support this e*Way.

2.5 Optional Example Files

The installation CD contains sample schemas located in the `samples\ewsiebeleim` directory. To use these schemas, you must load them onto your system using the following procedure. See [Sample Schema](#) on page 51 for descriptions of the sample schemas and instructions regarding their use.

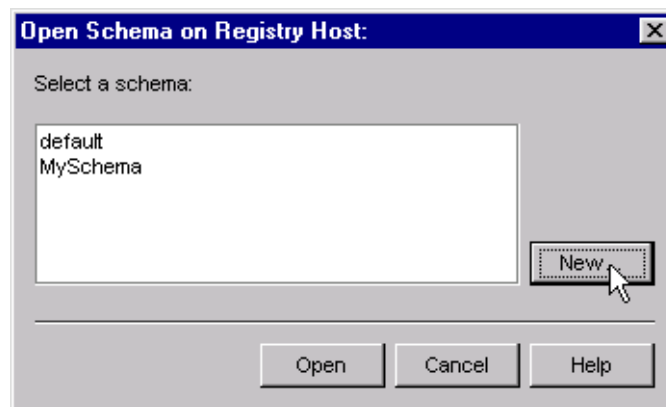
Note: *The Siebel EIM e*Way must be properly installed on your system before you can run the sample schema.*

2.5.1 Installation Procedure

To load a sample schema

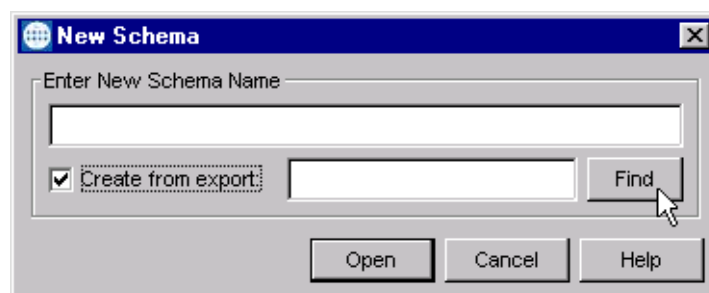
- 1 Invoke the **Open Schema** dialog box and select **New** (see Figure 7).

Figure 7 Open Schema Dialog



- 2 Type the name you want to give to the schema (for example, `EIM_Post.Sample`)
- 3 Select **Create from export** and navigate to the directory containing the sample schema by clicking the **Find** button (see Figure 8).

Figure 8 New Schema Dialog



- 4 Select the desired `*.zip` file and click **Open**.

Note: The schema installs with the host name **localhost** and control broker name **localhost_cb**. If you want to assign your own names, copy the file *.zip to a local directory and extract the files. Using a text editor, edit the file *.exp, replacing all instances of the name **localhost** with your desired name. Add the edited .exp file back into the .zip file.

2.5.2 Subdirectories and Files

The preceding procedure creates the following subdirectories and installs the following files within the \eGate\Server\registry\repository\

Siebel 2000

Table 8 Subdirectories and Files - Java_Siebel6_EIM_Extract

Subdirectories	Files
\runtime\	Java_Siebel_EIM_Extract2.ctl
\runtime\collaboration_rules\	FromSiebelEIM.class FromSiebelEIM.ctl FromSiebelEIM.java FromSiebelEIM.xpr FromSiebelEIM.xts FromSiebelEIMBase.class
\runtime\configs\oracle\	EIM_DB.cfg EIM_DB.sc
\runtime\configs\siebeleim\	EIMExtract.cfg EIMExtract.sc
\runtime\configs\stcewfile\	EIM_Eater.cfg EIM_Eater.sc Feeder_Trigger.cfg Feeder_Trigger.sc
\runtime\etd\	DbEimAccount.jar DbEimAccount.xbs DbEimAccount.xsc GenericBlob.jar GenericBlob.ssc GenericBlob.xsc GenericEvent.jar GenericEvent.ssc GenericEvent.xsc Siebel7DbEimAccount.jar Siebel7DbEimAccount.xbs Siebel7DbEimAccount.xsc

Table 9 Subdirectories and Files - Java_Siebel6_EIM_Post

Subdirectories	Files
\runtime\	Java_Siebel_EIM_Post2.ctl
\runtime\collaboration_rules\	ToSiebelEIM.class ToSiebelEIM.ctl ToSiebelEIM.java ToSiebelEIM.xpr ToSiebelEIM.xts ToSiebelEIMBase.class
\runtime\configs\oracle\	EIM_DB.cfg EIM_DB.sc EIM_DBinsert.cfg EIM_DBinsert.sc
\runtime\configs\siebeleim\	EIM_DBinsert.cfg EIM_DBinsert.sc eim_post.cfg eim_post.sc EIMExtract.cfg EIMExtract.sc
\runtime\configs\stcewfile\	EIM_Eater.cfg EIM_Eater.sc Feeder_Trigger.cfg Feeder_Trigger.sc
\runtime\etd\	AccountData.jar AccountData.xbs AccountData.xsc DbEimAccount.jar DbEimAccount.xbs DbEimAccount.xsc GenericBlob.jar GenericBlob.ssc GenericBlob.xsc GenericEvent.jar GenericEvent.ssc GenericEvent.xsc Siebel7DbEimAccount.jar Siebel7DbEimAccount.xbs Siebel7DbEimAccount.xsc
\sandbox\Administrator\collaboration_rules\	(empty)

Table 9 Subdirectories and Files - Java_Siebel6_EIM_Post

Subdirectories	Files
\sandbox\Administrator\etd\	AccountData.jar AccountData.xbs AccountData.xsc DbEimAccount.jar DbEimAccount.xbs DbEimAccount.xsc GenericBlob.jar GenericBlob.ssc GenericBlob.xsc GenericEvent.jar GenericEvent.ssc GenericEvent.xsc Siebel7DbEimAccount.jar Siebel7DbEimAccount.xbs Siebel7DbEimAccount.xsc
\sandbox\Administrator\etd\template\	AccountData.jar AccountData.ssc AccountData.xsc

Siebel 7

Table 10 Subdirectories and Files - Java_Siebel7_EIM_Extract

Subdirectories	Files
\runtime\	Java_Siebel7_EIM_Extract.ctl
\runtime\collaboration_rules\	FromSiebelEIM.class FromSiebelEIM.ctl FromSiebelEIM.java FromSiebelEIM.xpr FromSiebelEIM.xts FromSiebelEIMBase.class
\runtime\configs\oracle\	EIM_DB.cfg EIM_DB.sc
\runtime\configs\siebeleim\	EIMExtract.cfg EIMExtract.sc
\runtime\configs\stcewfile\	EIM_Eater.cfg EIM_Eater.sc Feeder_Trigger.cfg Feeder_Trigger.sc
\runtime\etd\	DbEimAccount.jar DbEimAccount.xbs DbEimAccount.xsc GenericBlob.jar GenericBlob.ssc GenericBlob.xsc GenericEvent.jar GenericEvent.ssc GenericEvent.xsc Siebel7DbEimAccount.jar Siebel7DbEimAccount.xbs Siebel7DbEimAccount.xsc
\sandbox\Administrator\configs\oracle\	(empty)
\sandbox\Administrator\configs\siebeleim\	(empty)
\sandbox\Administrator\etd\	DbEimAccount.jar DbEimAccount.xbs DbEimAccount.xsc GenericBlob.jar GenericBlob.ssc GenericBlob.xsc GenericEvent.jar GenericEvent.ssc GenericEvent.xsc Siebel7DbEimAccount.jar Siebel7DbEimAccount.xbs Siebel7DbEimAccount.xsc
\userlocks\configs\oracle\	(empty)
\sandbox\configs\siebeleim\	(empty)

Table 10 Subdirectories and Files - Java_Siebel7_EIM_Extract

Subdirectories	Files
\\sandbox\\etd\\	DbEimAccount.jar DbEimAccount.xbs DbEimAccount.xsc GenericBlob.jar GenericBlob.ssc GenericBlob.xsc GenericEvent.jar GenericEvent.ssc GenericEvent.xsc Siebel7DbEimAccount.jar Siebel7DbEimAccount.xbs Siebel7DbEimAccount.xsc

Table 11 Subdirectories and Files - Java_Siebel7_EIM_Post

Subdirectories	Files
\\runtime\\	Java_Siebel7_EIM_Post.ctf
\\runtime\\collaboration_rules\\	ToSiebelEIM.class ToSiebelEIM.ctf ToSiebelEIM.java ToSiebelEIM.xpr ToSiebelEIM.xts ToSiebelEIMBase.class
\\runtime\\configs\\oracle\\	EIM_DB.cfg EIM_DB.sc EIM_DBinsert.cfg EIM_DBinsert.sc
\\runtime\\configs\\siebeleim\\	EIM_DBinsert.cfg EIM_DBinsert.sc eim_post.cfg eim_post.sc EIMExtract.cfg EIMExtract.sc
\\runtime\\configs\\stcewfile\\	EIM_Eater.cfg EIM_Eater.sc Feeder_Trigger.cfg Feeder_Trigger.sc

Table 11 Subdirectories and Files - Java_Siebel7_EIM_Post

Subdirectories	Files
\runtime\etd\	AccountData.jar AccountData.xbs AccountData.xsc DbEimAccount.jar DbEimAccount.xbs DbEimAccount.xsc GenericBlob.jar GenericBlob.ssc GenericBlob.xsc GenericEvent.jar GenericEvent.ssc GenericEvent.xsc Siebel7DbEimAccount.jar Siebel7DbEimAccount.xbs Siebel7DbEimAccount.xsc
\sandbox\Administrator\collaboration_rules\	(empty)
\sandbox\Administrator\etd\	AccountData.jar AccountData.xbs AccountData.xsc DbEimAccount.jar DbEimAccount.xbs DbEimAccount.xsc GenericBlob.jar GenericBlob.ssc GenericBlob.xsc GenericEvent.jar GenericEvent.ssc GenericEvent.xsc Siebel7DbEimAccount.jar Siebel7DbEimAccount.xbs Siebel7DbEimAccount.xsc
\sandbox\Administrator\etd\template\	AccountData.jar AccountData.ssc AccountData.xsc

Table 11 Subdirectories and Files - Java_Siebel7_EIM_Post

Subdirectories	Files
\userlocks\etd\	AccountData.jar AccountData.xbs AccountData.xsc DbEimAccount.jar DbEimAccount.xbs DbEimAccount.xsc GenericBlob.jar GenericBlob.ssc GenericBlob.xsc GenericEvent.jar GenericEvent.ssc GenericEvent.xsc Siebel7DbEimAccount.jar Siebel7DbEimAccount.xbs Siebel7DbEimAccount.xsc
\userlocks\etd\template\	AccountData.jar AccountData.ssc AccountData.xsc

System Implementation

In this chapter we summarize the procedures required for implementing a working system incorporating the Java e*Way Intelligent Adapter for Siebel EIM. Please see the *e*Gate Integrator User's Guide* for additional information.

3.1 Overview

This e*Way provides a specialized transport component for incorporation in an operational schema. The schema also contains Collaborations, linking different data or Event types, and Intelligent Queues. Typically, other e*Way types also is used as components of the schema.

One or more sample schemas, included in the software package, are described at the end of this chapter. These can be used to test your system following installation and, if appropriate, as a template that you can modify to produce your own schema.

3.1.1 Pre-Implementation Tasks

Install the SeeBeyond Software

The first task is to install the SeeBeyond software as described in [Chapter 2](#).

Import the Sample Schema

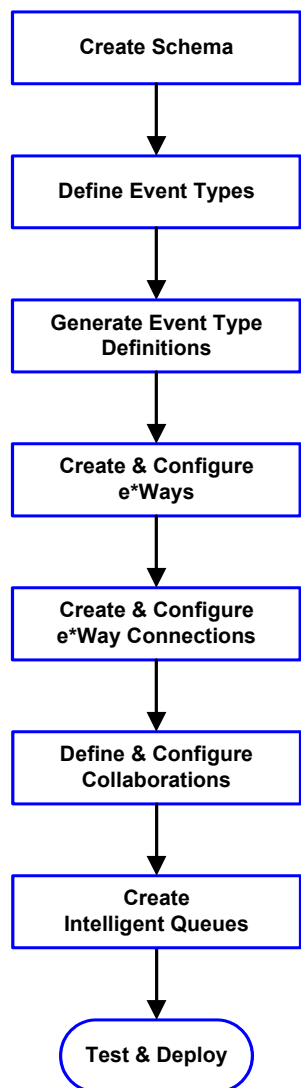
If you want to use the sample schema supplied with the e*Way, the schema files must be imported from the installation CD-ROM (see [Optional Example Files](#) on page 25).

Note: *It is highly recommended that you make use of the sample schemas to familiarize yourself with e*Way operation, test your system, and use as templates for your working schemas.*

3.1.2 Known Cautions

Note: *When DB2 UDB is employed as a Siebel RDMS, you must match the datatype **exactly**, because of the strict type binding enforced by DB2.*

3.1.3 Implementation Sequence



- 1 The first step is to create a new Schema—the subsequent steps apply only to this Schema (see [Creating a Schema](#) on page 35).
- 2 The second step is to define the Event Types you are transporting and processing within the Schema (see [Creating Event Types](#) on page 36).
- 3 Third, you need to associate the Event Types created in the previous step with Event Type Definitions (ETDs) derived from the applicable Business Rules (see [Generating Event Type Definitions](#) on page 37).
- 4 The fourth step is to create and configure the required e*Ways (see [Setting Up the e*Way](#) on page 55).
- 5 The fifth step is to configure the e*Way Connections (see [Creating e*Way Connections](#) on page 62).
- 6 Now you need to create Intelligent Queues to hold published Events (see [Creating Intelligent Queues](#) on page 48).
- 7 Next you need to define and configure the Collaborations between Event Types (see [Defining Collaborations](#) on page 49).
- 8 Finally, you must test your Schema. Once you have verified that it is working correctly, you may deploy it to your production environment.

3.1.4 Viewing e*Gate Components

Use the Navigator and Editor panes of the e*Gate Schema Designer to view the various e*Gate components. Note that you may only view components of a single schema at one time, and that all operations apply only to the current schema. All procedures in this chapter should be performed while displaying the **Components** Navigator pane. See the *e*Gate Integrator User's Guide* for a detailed description of the features and use of the Schema Designer.

3.2 Creating a Schema

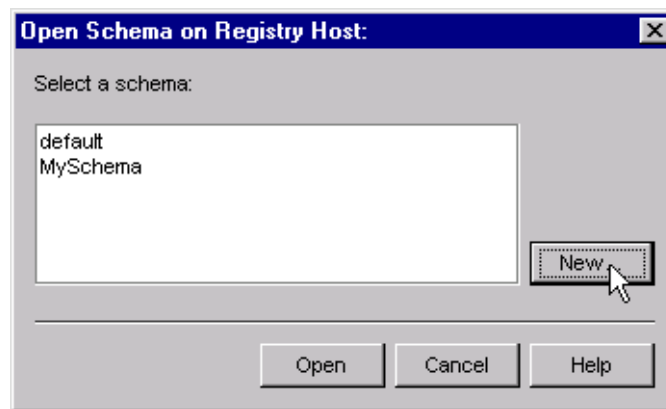
A schema is the structure that defines e*Gate system parameters and the relationships between components within the e*Gate system. Schemas can span multiple hosts.

Because all setup and configuration operations take place within an e*Gate schema, a new schema must be created, or an existing one must be started before using the system. Schemas store all their configuration parameters in the e*Gate Registry.

To select or create a schema

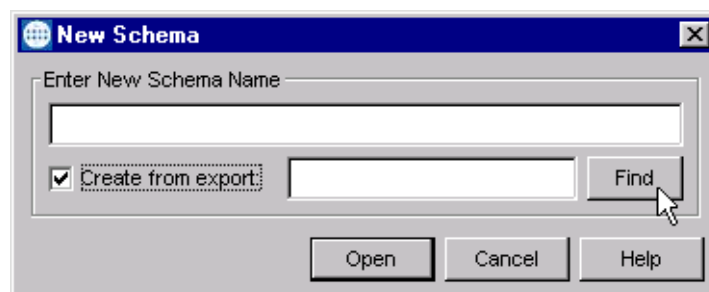
- 1 Invoke the **Open Schema** dialog box and **Open** an existing schema or click **New** to create a new schema.

Figure 9 Open Schema Dialog



- 2 Clicking **New** invokes the **New Schema** dialog box (Figure 10).

Figure 10 New Schema Dialog




- 3 Enter a new schema name and click **Open**.
- 4 The e*Gate Schema Designer then opens under your new schema name.
- 5 From the **Options** menu, click on **Default Editor** and select **Java**.
- 6 Select the **Components** tab, found at the bottom of the Navigator pane of the e*Gate Schema Designer window.
- 7 You are now ready to begin creating the necessary components for this new schema.

3.3 Creating Event Types

Within e*Gate, messages and/or packages of data are defined as Events. Each Event must be categorized into a specific Event Type within the schema.

To define the Event Types

- 1 In the e*Gate Schema Designer's Navigator pane, select the **Event Types** folder.
- 2 On the Palette, click the **New Event Type** button .
- 3 In the **New Event Type Component** box, enter the name for the input Event Type and click **Apply**. Use this method to create all required Event Types, for example:
 - ◆ **InboundEvent**
 - ◆ **ValidEvent**
 - ◆ **InvalidEvent**
- 4 After you have created the final Event Type, click **OK**.

3.4 Generating Event Type Definitions

3.4.1 Using the DBWizard ETD Builder

The DBWizard ETD Builder generates Java-enabled ETDs by connecting to external data sources and creating corresponding Event Type Definitions. The ETD Builder can create ETDs based on any combination of tables, stored procedures, or prepared SQL statements.

Field nodes are added to the ETD based on the tables in the external data source. Java method and parameter nodes are added to provide the appropriate JDBC functionality. For more information on the Java methods, refer to your JDBC developer's reference.

Note: For Japanese, Korean and Traditional Chinese operating systems, you must use English table names, and English column names within the tables (DBCS is not supported).

Please note that the DB Wizard is using ODBC standard data types. Specific data types that are not ODBC standard will not be supported. For example Oracle specific data type of PL/SQL TABLE.

To create a new ETD using the DBWizard

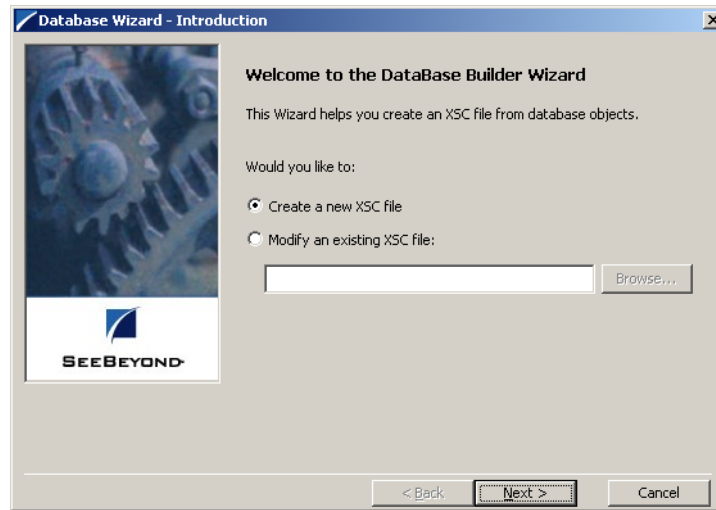
- 1 From the **Options** menu of the Schema Designer, choose **Default Editor**.
- 2 Verify that **Java** is selected, then click **OK**.
- 3 Click the **ETD Editor** button to launch the Java ETD Editor.
- 4 In the Java ETD Editor, click the **New** button to launch the New Event Type Definition window.
- 5 In the New Event Type Definition window, select the **DBWizard** and click **OK** to continue.

Figure 11 DBWizard Icon



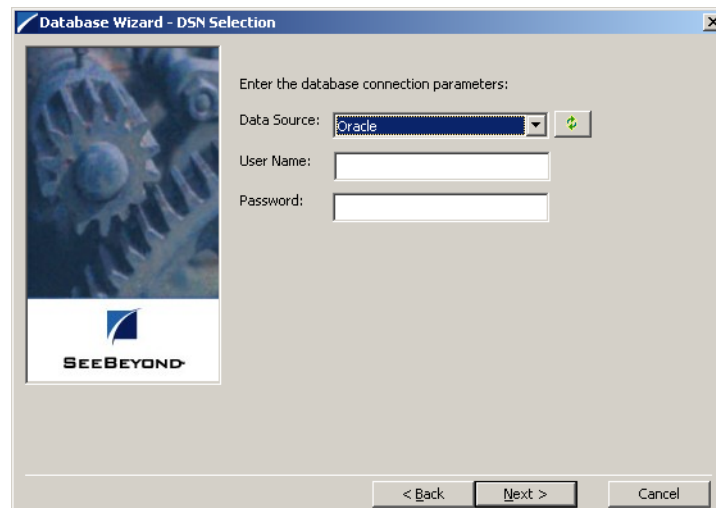
- 6 Enter the name of the new **.xsc** file you want to create or enter the name of the **.xsc** file you want to edit by browsing to its location.

Figure 12 Database Wizard - Introduction



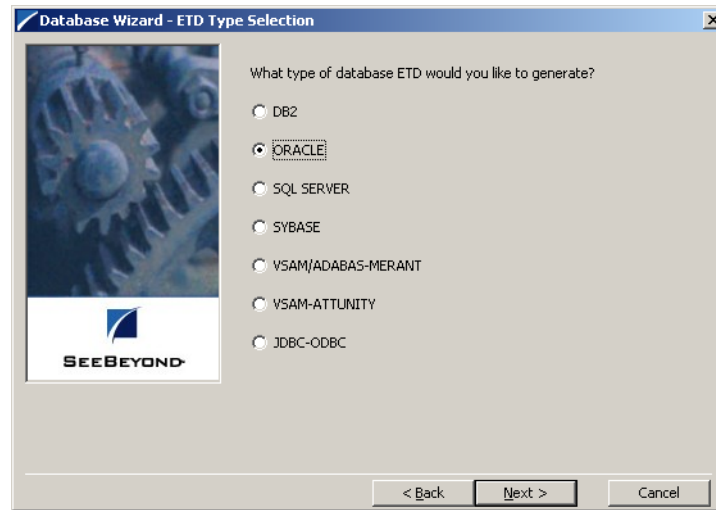
- 7 Select your **Data Source:** from the drop down list and enter your **User Name:** and **Password:**.

Figure 13 Database Wizard - DSN Selection



- 8 Select what type of database ETD you would like to generate. The data source you selected in the **Database Wizard - DSN Selection** window is the default. *Note: Do not change this unless instructed to do so by SeeBeyond personnel.*

Figure 14 Database Wizard - ETD Type Selection



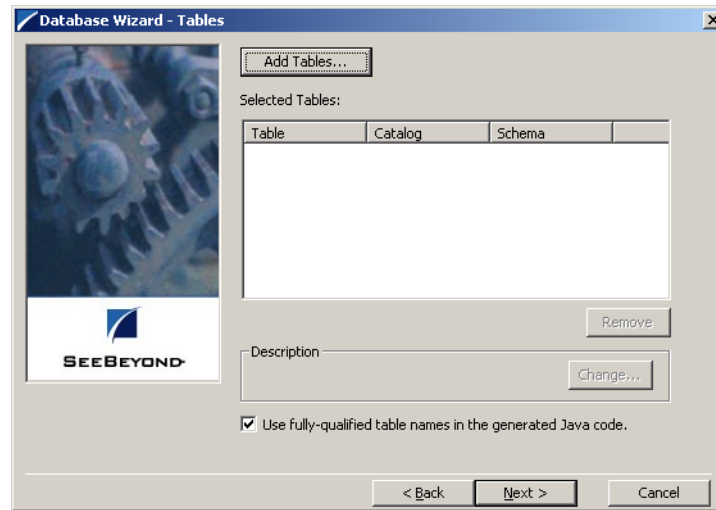
- 9 In the **Database Wizard - Object Selection** window, select any combination of **Tables**, **Views**, **Procedures**, or **Prepared Statements** you would like to include in your .xsc file.
- 10 Click **Next** to continue.

Figure 15 Database Wizard - Object Selection



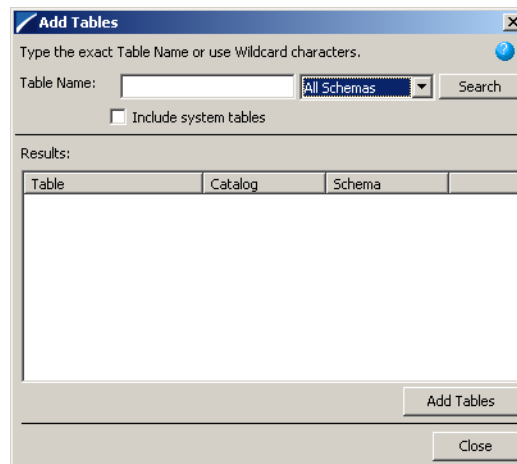
- 11 In the **Database Wizard - Tables** window, click **Add Tables**.

Figure 16 Database Wizard - Tables



- 12 In the **Add Tables** window, type the exact name of the database table or use wildcard characters to return table names.

Figure 17 Add Tables



- 13 To see a list of valid wildcard characters, click the round ball with a question mark located in its center.

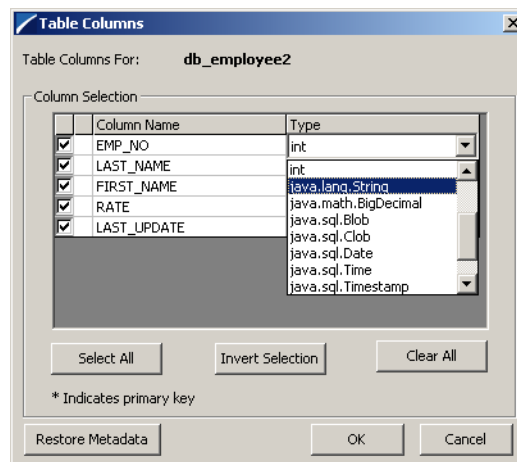
Note: When using the Oracle native ODBC driver with the DB Wizard to search for a table, view, or procedure, do not use `'_'` or `'%'`. Doing so will result in nothing being returned. Use the e*Way wildcard characters of `'?'`, and `'*'`. For example, searching for tables `"AB_CD"`, will return nothing. Use `"AB?CD"` or `"AB*CD"`.

Figure 18 Wildcards

Operator	Description	Example
*	any number of characters	DB*
?	any single character	DB?

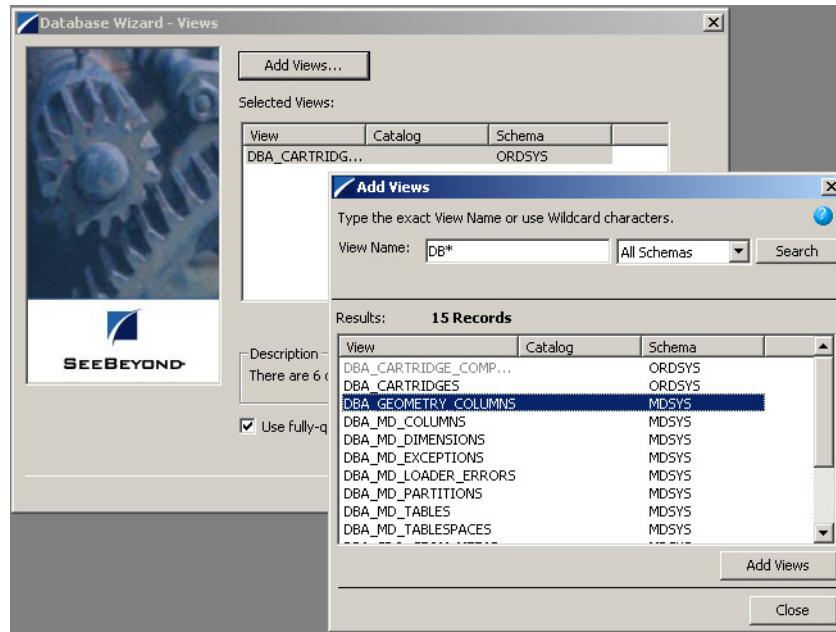
- 14 Select **Include System Tables** if you wish to include them and click **Search**. If your search was successful, you will see the results in the Results window. To select the name of the tables you wish to add to your `.xsc`, double click on the table name or highlight the table names and click **Add Tables**. You may also use adjacent selections or nonadjacent selections to select multiple table names. When you have finished, click **Close**.
- 15 In the **Database Wizard - Tables** window, review the tables you have selected. If you would like to change any of the tables columns you have selected, click **Change**.
- 16 In the **Columns Selection** window, you can select or deselect your table columns. 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. Once you have completed your choices, click **OK**.

Figure 19 Columns Selection



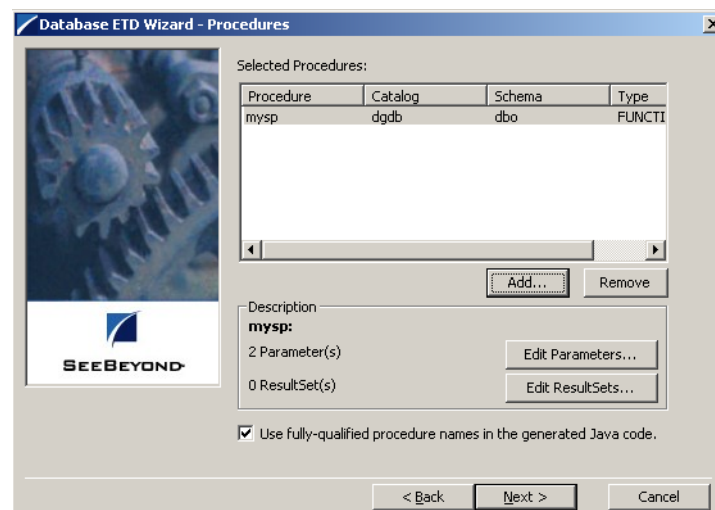
- 17 In the **Database Wizard - Tables** window, review the tables you have selected. If you do not want to use fully-qualified table names in the generated Java code, click to clear the check box and click **Next** to continue.
- 18 If you selected **Views** on the **Database Wizard - Object Selection** window, you are presented with the **Database Wizard - Views** window. Follow steps 9 - 17 to select and add views to your `.xsc`. Views are read-only.

Figure 20 Database Wizard - Views



- 19 If you selected **Procedures** on the **Database Wizard - Object Selection** window, you are presented with the **Database Wizard - Procedures** window. Follow steps 9 - 15 to select and add **Procedures** to your .xsc. If you do not want to use fully-qualified procedure names in the generated Java code, click to clear the check box and click **Next** to continue.

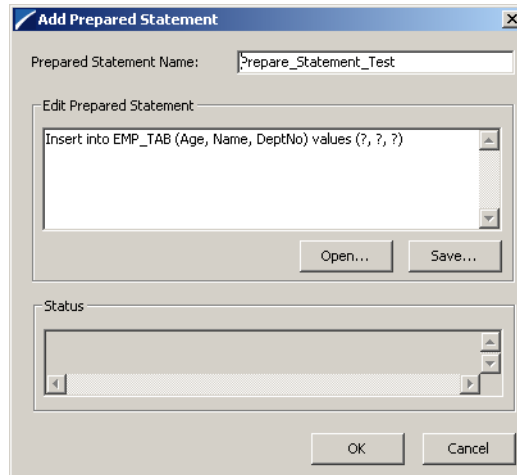
Figure 21 Database Wizard - Procedures



- 20 If you selected **Prepared Statements** on the **Database Wizard - Object Selection** window, you are presented with the **Database Wizard - Prepared Statement** window. To add **Prepared Statements** to your .xsc. complete the following steps:
- A Click **Add** to add a new prepared statement

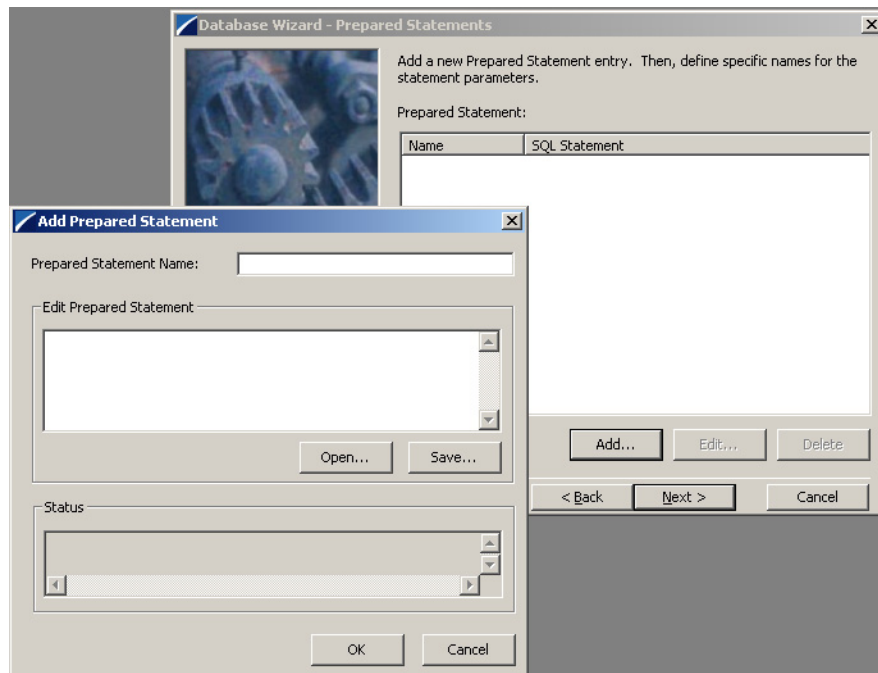
- B Enter a prepared SQL statement.
- C Enter the **Prepared Statement Name** to be used by the statement.
- D Use the **Open...** or **Save...** button to open pre-existing statements or save the current one. See Figure 22.

Figure 22 Add Prepared Statement



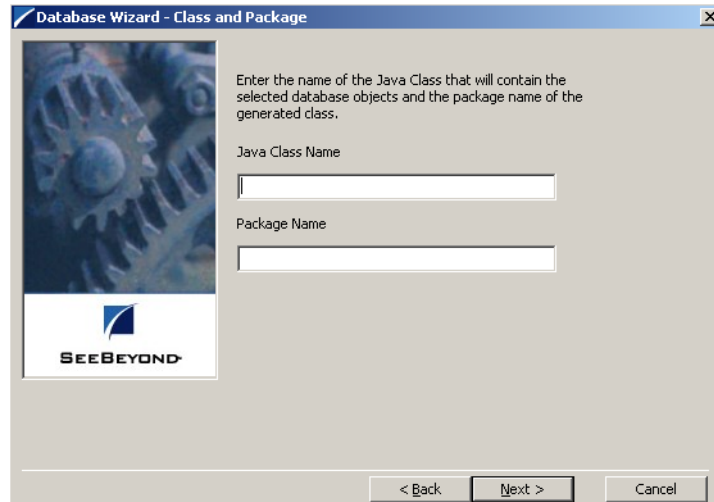
- E Click **OK** to return to the **Database Wizard - Prepared Statements** window.
- 21 Repeat steps A–E to add additional prepared statements or click **Next** to continue.

Figure 23 Database Wizard - Prepared Statements



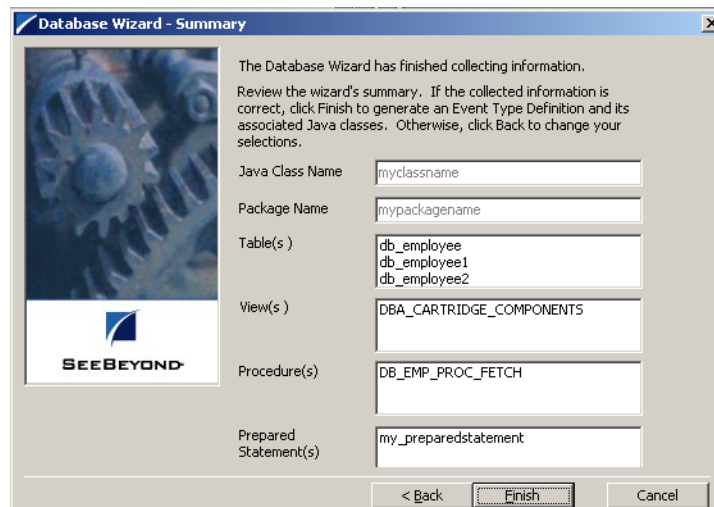
- 22 Enter the **Java Class Name** that will contain the selected tables and/or procedures and the **Package Name** of the generated classes.

Figure 24 Database Wizard - Class and Package



- 23 View the summary of the database wizard information and click **Finish** to begin generating the ETD.

Figure 25 Database Wizard - Summary



3.4.2 The Generated ETDs

The DataBase Wizard ETD builder can create three editable Event Type Definitions (ETDs) and one non-editable Event Type Definition (ETD). These types of ETDs can also be combined with each other. The four types of ETDs are:

- **Table ETD**

The table ETD contains fields for each of the columns in the selected table as well as the methods required to exchange data with the external data source. To edit this type of ETD, you will need to open the `.xsc` file in the DataBase Wizard.

- **View ETD**

The view ETD contains selected columns from selected tables. View ETDs are read-only.

- **Stored Procedure ETD**

The stored procedure ETD contains fields which correspond to the input and output fields in the procedure. To edit this type of ETD, you will need to open the `.xsc` file in the DataBase Wizard.

- **Prepared Statement ETD**

The prepared statement ETD contains a result set for the prepared statement. To edit this type of ETD, you will need to open the `.xsc` file in the DataBase Wizard.

3.4.3 Editing an Existing ETD Using the Database Wizard

If you choose to edit an existing `.xsc` file that you have created using the Database Wizard, do the following:

- 1 From the **Options** menu of the 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 Tool menu click **File** and click **New**.
- 5 From the **New Event Type Definition** window, select **DBWizard** and click **OK**.
- 6 On the Database Wizard - Introduction window, select **Modify an existing XSC file:** and browse to the appropriate `.xbs` file that you would like to edit.

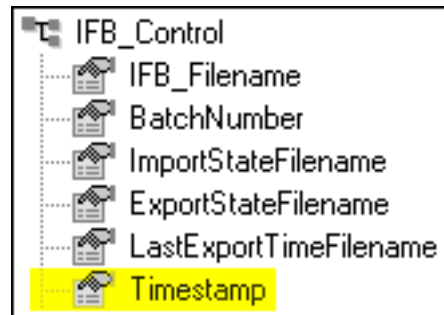
You are now able to edit your `.xsc` file.

Note: *When you add a new element type to your existing `.xsc` file, you must reselect any pre-existing elements or you will lose them when the new `.xsc` file is created.*

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

3.4.4 Using the Timestamp Node

Figure 26 IFB_Control - Timestamp Node



If you want to obtain records from the database within a certain time frame, you can use the **Timestamp** attribute in the collaboration, as follows:

- 1 Get the last export time, using the method from the ETD.
- 2 Get the current time from the database. The SQL commands are as follows:

- ♦ **DB2**

```
"select CURRENT_TIMESTAMP FROM SYSIBM.SYSDUMMY1"
```

- ♦ **Oracle**

```
"alter session set NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS'"
"select sysdate from dual"
```

- ♦ **SQL Server**

```
"select {fn NOW()}"
```

- 3 Write a command into the IFB file using the last export time and the current time, such as the following:

```
EXPORT MATCHES = S_ORG_EXT, (LAST_UPD >= '2002-04-25 09:38:36' AND
LAST_UPD < '2002-06-21 10:38:36')
```

An example is given in the [EIM_Extract](#) sample schema.

Note: *Timestamps are obtained from the Siebel Database Server. If different Siebel components are installed on different systems (within the same overall installation), they must be synchronized for this feature to work properly.*

Example Synchronization Problem

As an example of the problem alluded to in the above note, consider having the Siebel Server on one platform and a Database Server on another platform. The timestamp used by Siebel to populate the database will be later than the one returned by the Database Server. If the Post schema is running *immediately* after the Extract schema, the Extract program will *not* be able to pick up new messages. If the Extract schema will be run again in a later time (for example, if the e*Way is running in poll mode), however, the new messages will be eventually picked up. Synchronizing the timestamps between the Siebel components will avoid this problem.

3.5 Assigning ETDs to Event Types

After you have created the e*Gate system's ETD files, you can assign them to Event Types you have already created.

To assign ETDs to Event Types


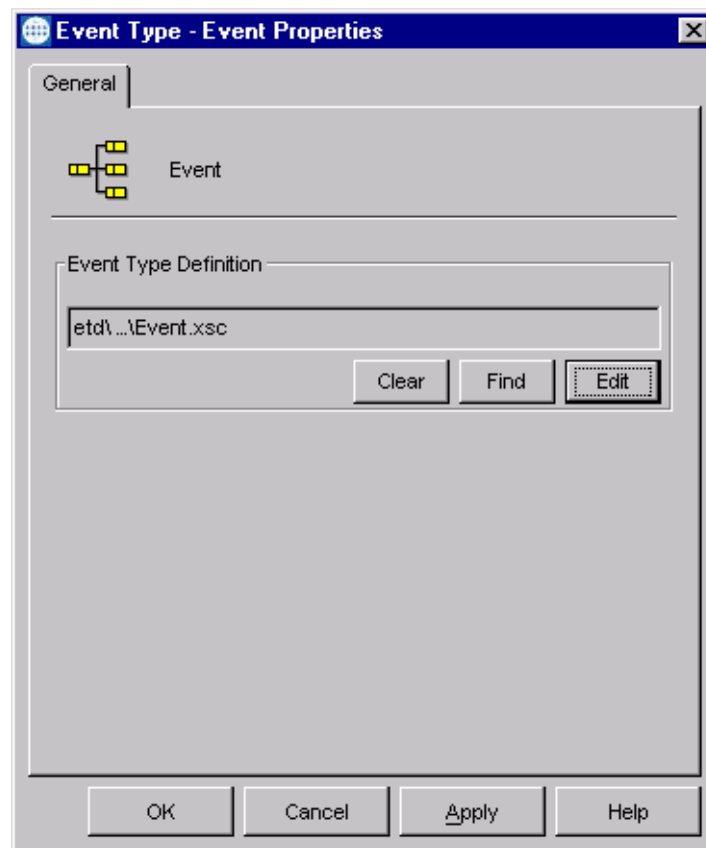
- 1 In the Schema Designer window, select the **Event Types** folder in the Navigator/Components pane.
- 2 In the Editor pane, select one of the Event Types you created.
- 3 Right-click on the Event Type and select **Properties** (or click  in the toolbar). The Event Type Properties dialog box appears. See Figure 27.

Figure 27 Event Type Properties Dialog Box



- 4 Under Event Type Definition, click **Find**.
The Event Type Definition Selection dialog box appears; it is similar to the Windows Open dialog box.
- 5 Open the **etd** folder, then select the desired file name (.xsc). Note that there may be an intervening sub-folder.
- 6 Click **Select**. The file populates the Event Type Definition field.

- 7 To save any work in the properties dialog box, click **Apply** to enter it into the system.
- 8 When finished assigning ETDs to Event Types, click **OK** to close the properties dialog box and apply all the properties.

Each Event Type is associated with the specified Event Type Definition.

3.6 Creating Intelligent Queues

IQs are components that provide nonvolatile storage for Events within the e*Gate system as they pass from one component to another. IQs are *intelligent* in that they are more than just a “holding tank” for Events. They actively record information about the current state of Events.

Each schema must have an IQ Manager before you can add any IQs to it. You must create at least one IQ per schema for published Events within the e*Gate system. Note that e*Ways that publish Events externally do not need IQs.

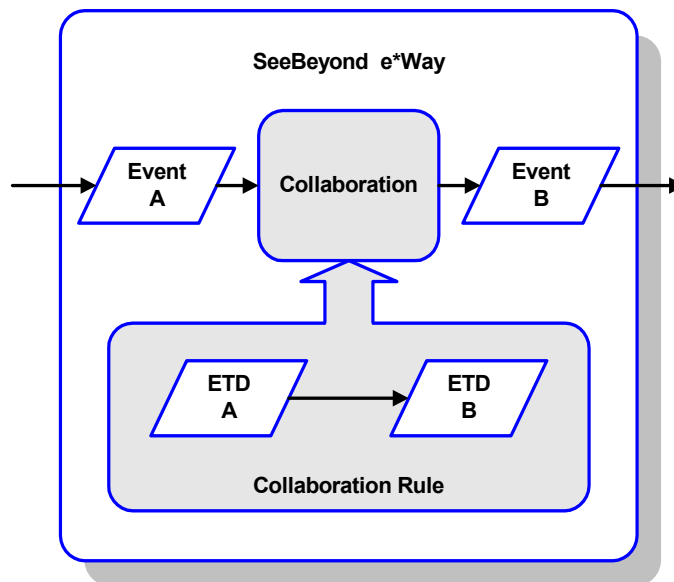
For more information on how to add and configure IQs and IQ Managers, see the *e*Gate Integrator System Administration and Operations Guide*. See the *e*Gate Integrator Intelligent Queue Services Reference Guide* and the *SeeBeyond JMS Intelligent Queue User’s Guide* for complete information on working with IQs.

3.7 Defining Collaborations

After you have created the required Event Type Definitions, you must define a Collaboration to transform the incoming Event into the desired outgoing Event.

Collaborations are e*Way components that receive and process Event Types, then forward the output to other e*Gate components. Collaborations consist of the Subscriber, which “listens” for Events of a known type or from a given source, and the Publisher, which distributes the transformed Event to a specified recipient. The same Collaboration cannot be assigned to more than one e*Gate component.

Figure 28 Collaborations



3.7.1 The Java Collaboration Rules Editor

Java Collaborations are defined using the e*Gate Java Collaboration Rules Editor. Note that the Java Collaboration environment supports multiple source and destination ETDs. The file extension for Java Collaboration Rules is **.xpr**. See the *e*Gate Integrator User's Guide* for descriptions of the Java Collaboration Rules Editor and its use.

3.7.2 Subcollaboration Rules

New methods are provided to allow you to use a Collaboration Rule as a parent or child of another Collaboration Rule. This allows you to insulate connectivity from transformation by separating out transformation-specific Collaboration Rules and maintaining them as individual units. See the *e*Gate Integrator User's Guide* for detailed information on Subcollaborations.

3.8 Auditing

Error counts and totals are returned to the e*Way through variables. When a non-zero return code is given, these errors are written to the standard e*Way error logs.

3.8.1 Error logging—Inbound to Siebel

Errors are logged in the Siebel interface tables, a custom file is created by the Siebel EIM e*Way, and the standard error logs are supplied by the e*Gate components. The Siebel EIM e*Way monitors the status of EIM by capturing the task state for the initiated interface. The state changes to **Completed**, thus signaling the end of the EIM process.

The Siebel EIM e*Way interrogates the status of the **IF_ROW_STAT** column in the interface table where the batch number equals the records just loaded. The Siebel EIM e*Way extracts all records from the interface tables where the **IF_ROW_STAT** does not equal **Imported**, and sends an alert to the Administrator that manual intervention is required.

3.8.2 Error logging—Outbound from Siebel

Although Siebel lacks an alert notification facility to notify administrators, it does log the success or failure of each row during the export process. If the exported data cannot be extracted from the database, a message is recorded in the e*Way log and a Notification is sent to the Schema Manager.

3.8.3 Journaling

Errors encountered during normal e*Gate processing are addressed by normal functionality. Each Logical Unit of Work (LUW) that encounters errors during translation is placed into a journal for later viewing, correction and reprocessing.

3.9 Sample Schema

Sample implementations are available in the `\samples\ewsiebeleim` directory of the e*Gate CD-ROM.

- **Java_Siebel6_EIM_Extract**
Siebel 2000-to-e*Gate example.
- **Java_Siebel6_EIM_Post**
e*Gate-to-Siebel 2000 example.
- **Java_Siebel7_EIM_Extract**
Siebel 7-to-e*Gate example.
- **Java_Siebel7_EIM_Post**
e*Gate-to-Siebel 7 example.

These samples can be used to test your system following installation and, if appropriate, as templates that you can modify to produce your own schemas. See [Optional Example Files](#) on page 25 for installation instructions.

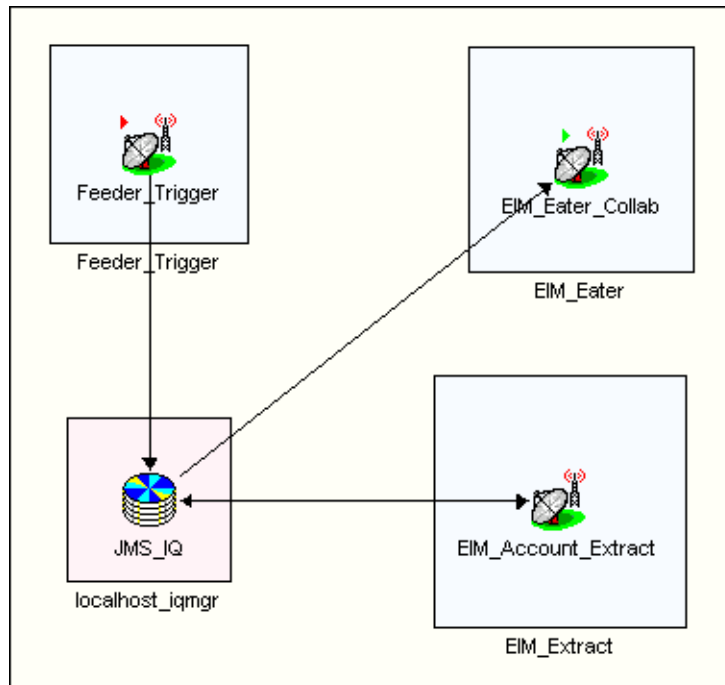
The corresponding schemas for Siebel 2000 and Siebel 7 are identical in architecture, but are different in detail to match the different table structures in the two Siebel versions.

3.9.1 EIM_Extract

For an explanation of the operational details, including annotated code, see [Siebel to e*Gate](#) on page 76.

Architecture

Figure 29 EIM_Extract Schema



Operation

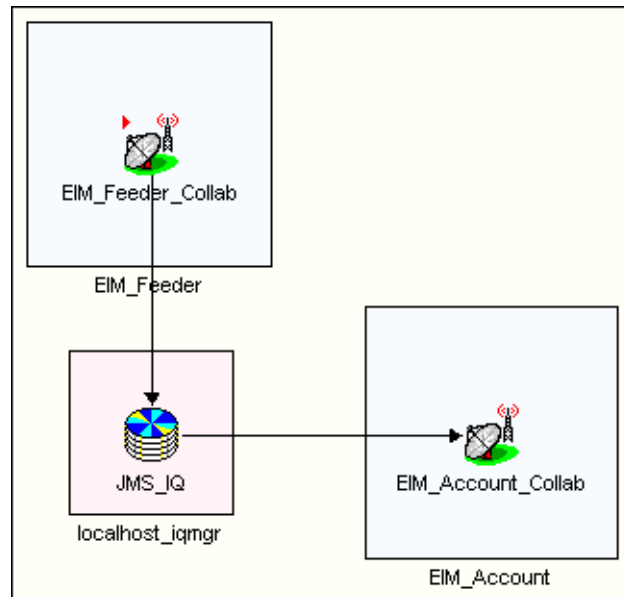
- 1 Upon receiving a trigger from **Feeder_Trigger**, the Siebel EIM e*Way, **EIM_Extract**, generates an IFB Control File, and sends it to Siebel.
- 2 The e*Way then invokes the Siebel EIM process.
- 3 Siebel EIM then copies data from the Siebel Base Tables into the Interface Tables, as prescribed by the IFB Control File.
- 4 **EIM_Extract** extracts the data from the interface tables and maps it into the correct Event Type Definition, following a pre-defined Collaboration.
- 5 The data is then passed to **EIM_Eater** and is available as a file.

3.9.2 EIM_Post

For an explanation of the operational details, including annotated code, see [e*Gate to Siebel](#) on page 81.

Architecture

Figure 30 EIM_Post Schema



Operation

- 1 The File e*Way **EIM_Feeder** sends data to a JMS Intelligent Queue.
- 2 The Siebel EIM e*Way, **EIM_Account**, processes the information following a Collaboration previously created using structural information extracted from Siebel, and inserts validated rows into the Siebel Interface Tables.
- 3 An IFB control file is built and sent to the Siebel system for use by the EIM in populating the Base Tables with data from the Interface Tables.
- 4 **EIM_Account** initiates the Siebel Enterprise Integration Manager (EIM) to load the data from the Interface Tables into the appropriate Siebel Base Tables.
- 5 Upon completion of the EIM process, **EIM_Account** interrogates the **IF_ROW_STAT** column in the Siebel Interface Tables to verify the correct transfer of data to the Base Tables. All records transferred successfully are deleted from the Interface Tables.

e*Way Setup

This chapter describes the procedures required to customize the SeeBeyond Java e*Way Intelligent Adapter for Siebel EIM to operate within your production system.

4.1 Overview

After creating a schema, you must instantiate and configure the e*Way Intelligent Adapter for Siebel EIM to operate within the schema. A wide range of setup options allow the e*Way to conform to your system's operational characteristics and your facility's operating procedures.

The topics discussed in this chapter are:

[Setting Up the e*Way](#) on page 55

[Creating e*Way Connections](#) on page 62

[Managing e*Way Connections](#) on page 66

[Using the e*Way Configuration Editor](#) on page 68

[Troubleshooting the e*Way](#) on page 71

4.2 Setting Up the e*Way

Note: The e*Gate Schema Designer GUI runs only on the Windows operating system.

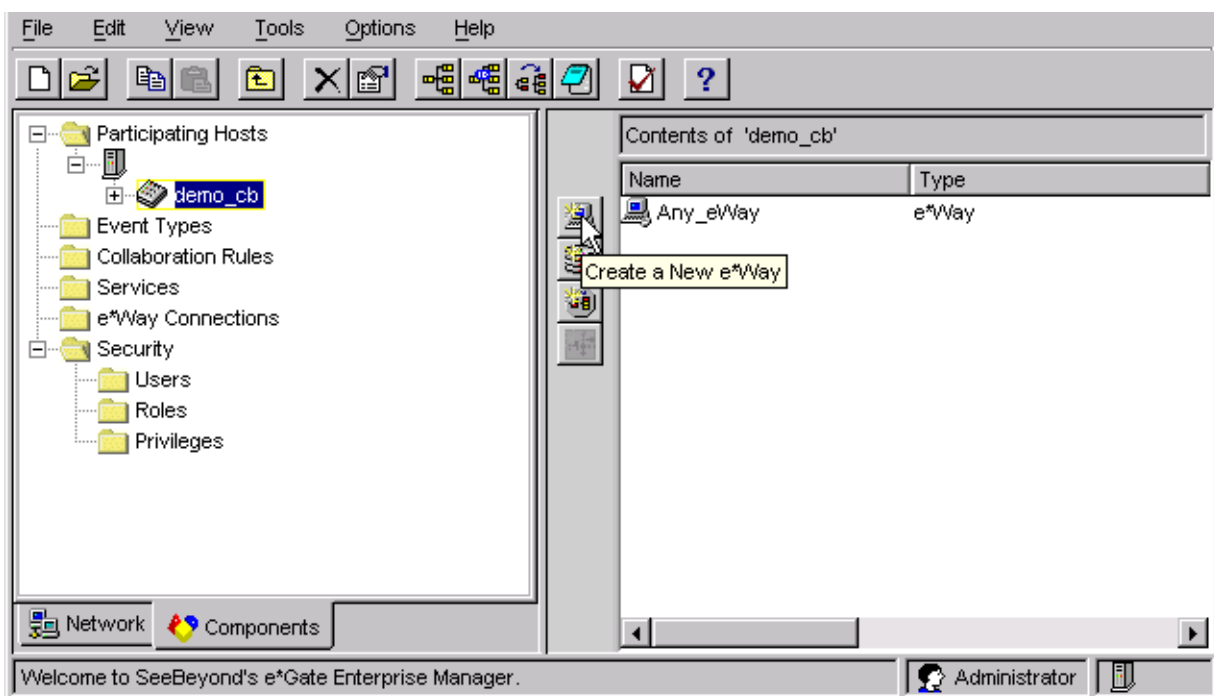
4.2.1 Creating the e*Way

The first step in implementing an e*Way is to define the e*Way component using the e*Gate Schema Designer.

To create an e*Way

- 1 Open the schema in which the e*Way is to operate.
- 2 Select the e*Gate Schema Designer Navigator's **Components** tab.
- 3 Open the host on which you want to create the e*Way.
- 4 Select the Control Broker you want to manage the new e*Way.

Figure 31 e*Gate Schema Designer Window (Components View)



- 5 On the Palette, click **Create a New e*Way**.
- 6 Enter the name of the new e*Way, then click **OK**.
- 7 All further actions are performed in the e*Gate Schema Designer Navigator's **Components** tab.

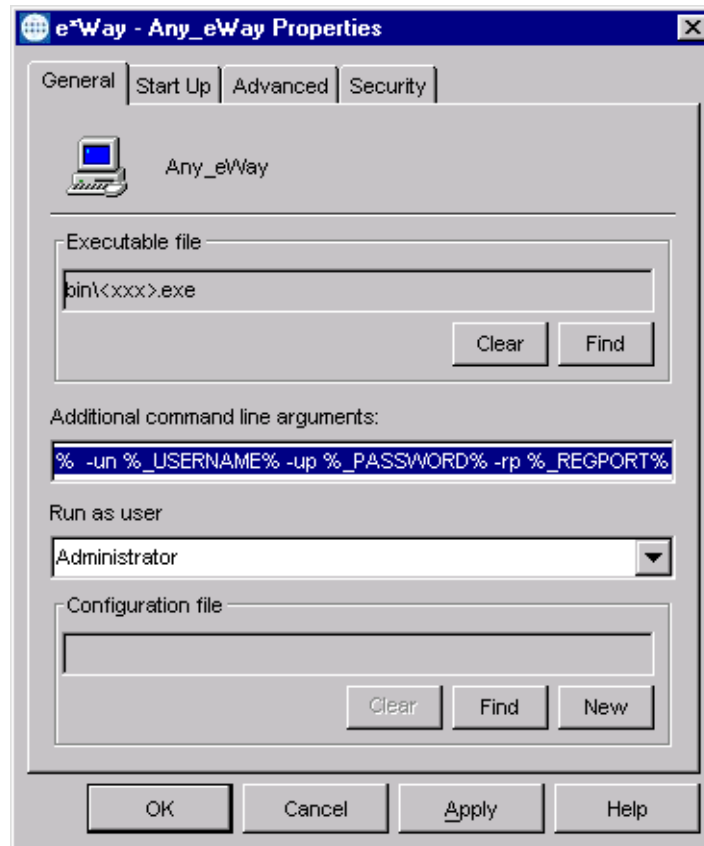
4.2.2 Modifying e*Way Properties

To modify any e*Way properties

- 1 Right-click on the desired e*Way and select **Properties** to edit the e*Way's properties. The properties dialog opens to the **General** tab (shown in Figure 32).

Note: The figures are generic—the executable and default configuration files used by this e*Way are listed in **e*Way Components** on page 16.

Figure 32 e*Way Properties (General Tab)



- 2 Make the desired modifications, then click **OK**.

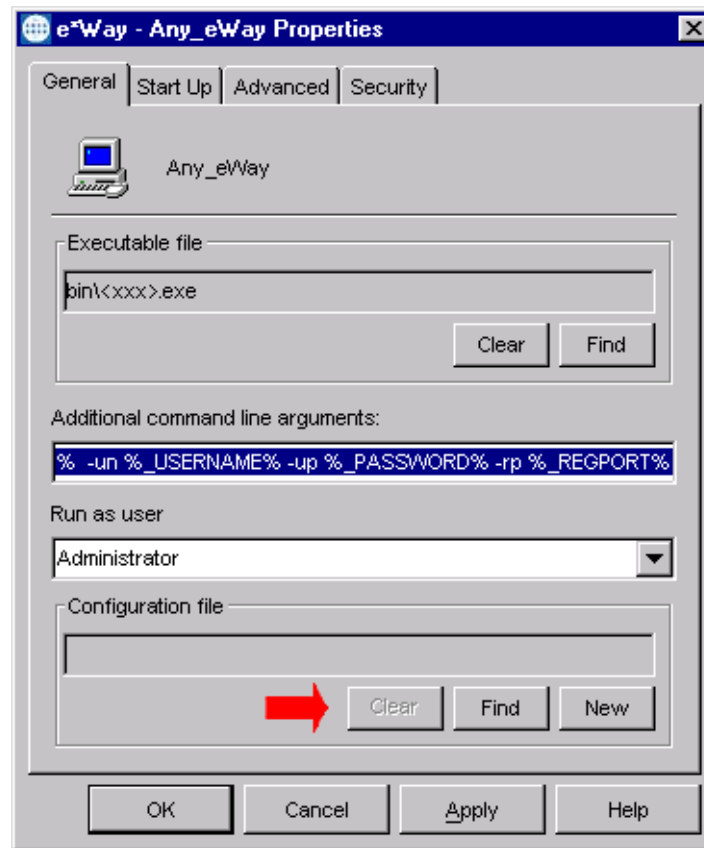
4.2.3 Configuring the e*Way

The e*Way's inherent configuration parameters are stored in an ASCII text file with a .def extension. The e*Way Editor provides a simple graphical interface for viewing and changing those parameters to create a working configuration (.cfg) file.

To change e*Way configuration parameters

- 1 In the e*Gate Schema Designer's Component editor, select the e*Way you want to configure and display its properties.

Figure 33 e*Way Properties - General Tab



- 2 Under **Executable File**, click **Find** to locate `stceway.exe`.
- 3 Under **Configuration File**, click **New** to create a new file or **Find** to select an existing configuration file. If you select an existing file, an **Edit** button appears; click the button to edit the currently selected file.
- 4 You now are in the e*Way Configuration Editor (see [Using the e*Way Configuration Editor](#) on page 68). The e*Way's configuration parameters are described in [Chapter 6](#).

4.2.4 Changing the User Name

Like all e*Gate executable components, e*Ways run under an e*Gate user name. By default, all e*Ways run under the **Administrator** user name. You can change this if your site's security procedures so require.

To change the user name

- 1 Display the e*Way's properties dialog.
- 2 On the **General** tab, use the **Run as user** list to select the e*Gate user under whose name you want this component to run.

See the *e*Gate Integrator System Administration and Operations Guide* for more information on the e*Gate security system.

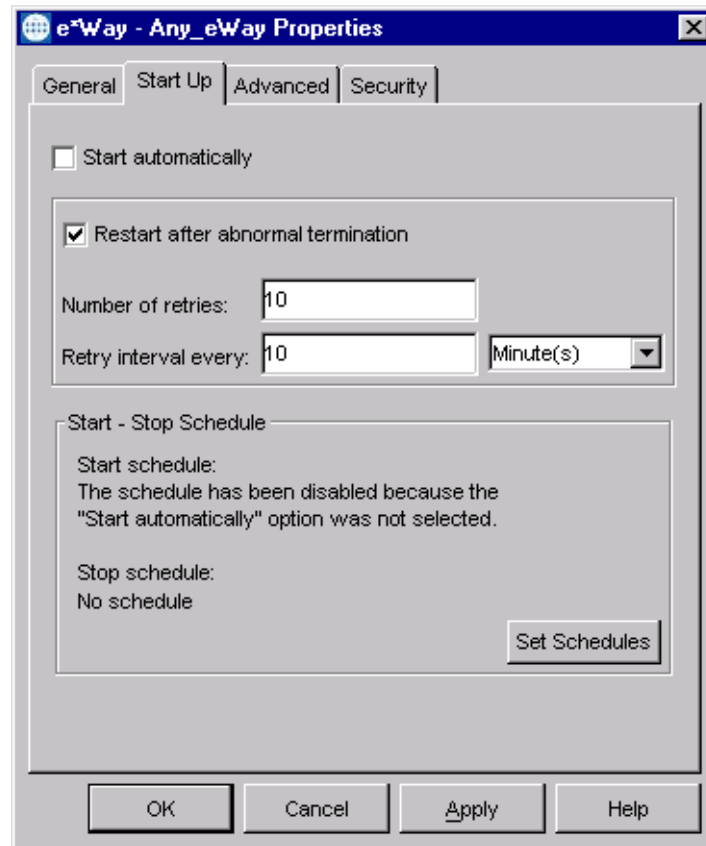
4.2.5 Setting Startup Options or Schedules

SeeBeyond e*Ways can be started or stopped by any of the following methods:

- The Control Broker can start the e*Way automatically whenever the Control Broker starts.
- The Control Broker can start the e*Way automatically whenever it detects that the e*Way terminated execution abnormally.
- The Control Broker can start or stop the e*Way on a schedule that you specify.
- Users can start or stop the e*Way manually using an interactive monitor.

You determine how the Control Broker starts or shuts down an e*Way using options on the e*Way properties **Start Up** tab (see Figure 34). See the *e*Gate Integrator System Administration and Operations Guide* for more information about how interactive monitors can start or shut down components.

Figure 34 e*Way Properties (Start-Up Tab)



To set the e*Way's startup properties

- 1 Display the e*Way's properties dialog.
- 2 Select the **Start Up** tab.
- 3 To have the e*Way start automatically when the Control Broker starts, select the **Start automatically** check box.
- 4 To have the e*Way start manually, clear the **Start automatically** check box.
- 5 To have the e*Way restart automatically after an abnormal termination:
 - A Select **Restart after abnormal termination**.
 - B Set the desired number of retries and retry interval.
- 6 To prevent the e*Way from restarting automatically after an abnormal termination, clear the **Restart after abnormal termination** check box.
- 7 Click **OK**.

Note: Clearing the **Start automatically** check box will delete any existing schedules (a warning dialog will be displayed first).

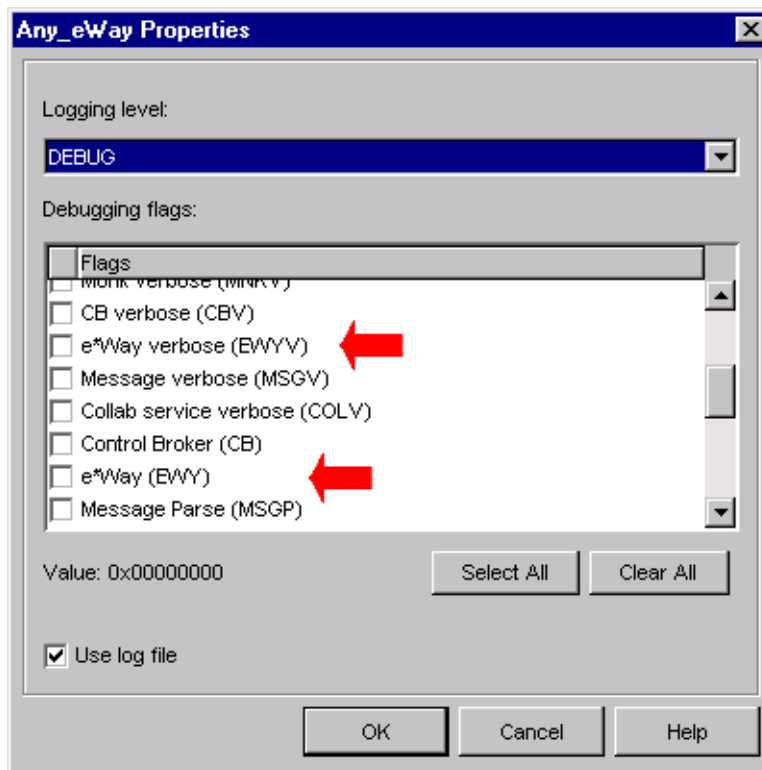
4.2.6 Activating or Modifying Logging Options

Logging options enable you to troubleshoot problems with the e*Way and other e*Gate components.

To set the e*Way debug level and flag

- 1 Display the e*Way's Properties dialog.
- 2 Select the **Advanced** tab.
- 3 Click **Log**. The dialog window appears, as shown in Figure 35.

Figure 35 e*Way Properties (Advanced Tab - Log Option)



- 4 Select **DEBUG** for the **Logging level**.
- 5 Select either **e*Way (EWY)** or **e*Way Verbose (EWYV)** for the **Debugging flag**. Note that the latter has a significant impact on system performance.
- 6 Click **OK**.

The other options apply to other e*Gate components and are activated in the same manner. See the *e*Gate Integrator Alert and Log File Reference* for additional information concerning log files, logging options, logging levels, and debug flags.

4.2.7 Activating or Modifying Monitoring Thresholds

Monitoring thresholds enable you to monitor the throughput of the e*Way. When the monitoring thresholds are exceeded, the e*Way sends a Monitoring Event to the Control Broker, which routes it to the Schema Manager and any other configured destinations.

- 1 Display the e*Way's properties dialog.
- 2 Select the **Advanced** tab.
- 3 Click **Thresholds**.
- 4 Select the desired threshold options and click **OK**.

See the *e*Gate Integrator Alert and Log File Reference* for more information concerning threshold monitoring, routing specific notifications to specific recipients, or for general information about e*Gate's monitoring and notification system.

4.3 Creating e*Way Connections

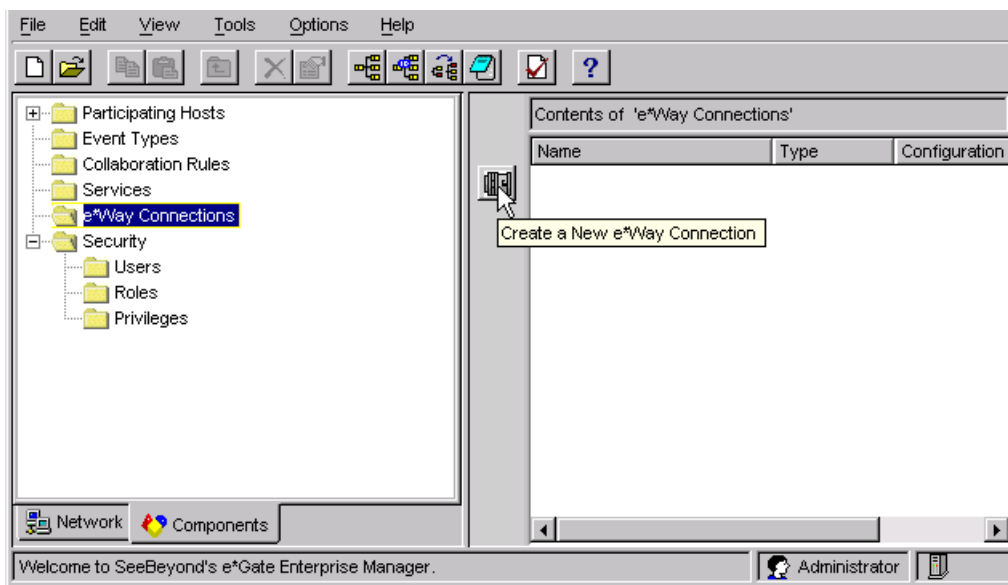
The e*Way Connections are created and configured in the Schema Designer. The e*Way Connection's configuration parameters are described in [Siebel EIM e*Way Connections](#) on page 112.

Note: The e*Gate Schema Designer GUI runs only on the Windows operating system.

To create and configure the e*Way Connections

- 1 In the Schema Designer's Component editor, select the **e*Way Connections** folder.

Figure 36 Schema Designer - e*Way Connections Folder (1)




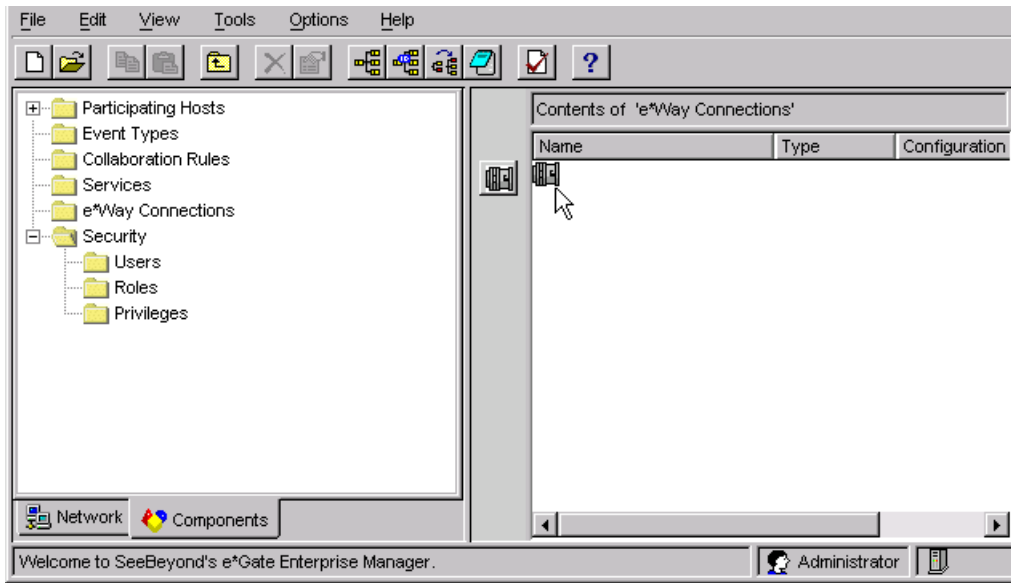
- 2 On the Palette, click the **Create a New e*Way Connection** button , which opens the New e*Way Connection Component dialog box.

Figure 37 New e*Way Connection Component Dialog Box



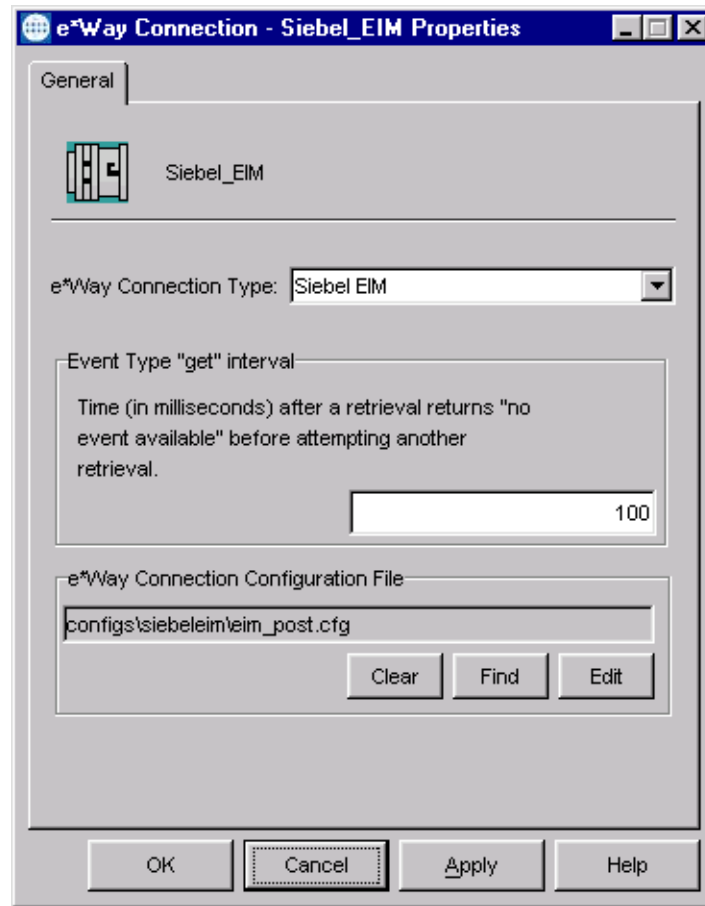
- 3 Enter a name for the e*Way Connection and click **OK**. The new e*Way Connection appears in the Schema Designer Contents pane.

Figure 38 Schema Designer - e*Way Connections Folder (2)



- 4 Right-click the new e*Way Connection icon and select **Properties** to open the e*Way Connection Properties dialog box.

Figure 39 e*Way Connection Properties Dialog Box



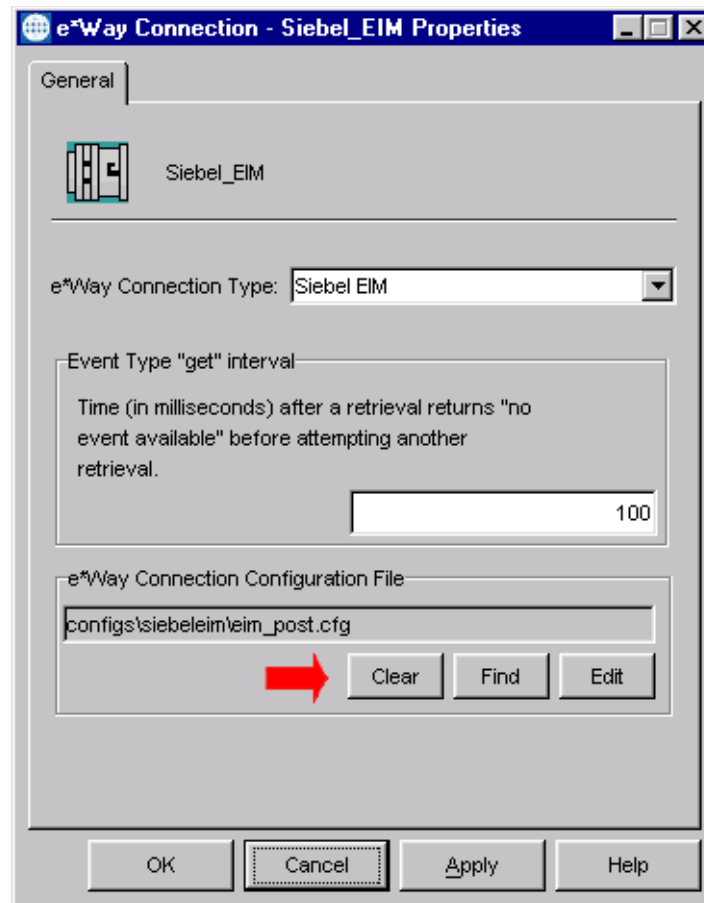
- 5 From the e*Way Connection Type drop-down box, select **Siebel EIM** (or one of the Database e*Way connections, as is appropriate).
- 6 Enter the Event Type *get* interval in the dialog box provided (optional).
- 7 Click **New** to start the e*Way Connection Configuration File Editor, where you can create a new e*Way Connection Configuration File.

To modify the e*Way Connections

- 1 In the e*Gate Schema Designer's Component editor, select the e*Way you want to reconfigure and display its properties.

Note: The executable and default configuration files used by this e*Way are listed in **e*Way Components** on page 16. The configuration files for the sample schemas are listed in **Optional Example Files** on page 25.

Figure 40 e*Way Connection Properties Dialog Box



- 2 Under **Configuration File**, click **New** to create a new file or **Find** to select an existing configuration file. If you select an existing file, an **Edit** button appears; click the button to edit the currently selected file.
- 3 You are now in the e*Way Configuration Editor (see **Using the e*Way Configuration Editor** on page 68). The e*Way Connection's configuration parameters are described in **Siebel EIM e*Way Connections** on page 112.

Note: You must restart the e*Way after modifying the e*Way connection.

4.4 Managing e*Way Connections

The Connection Manager allows you to define the functionality of your e*Way connection. You can:

- Select when to initiate an e*Way connection
- Select when to close the e*Way connection and disconnect
- Select the status of your e*Way connection
- Have the Collaboration call the **OnConnectionDown** method when the connection fails

The Connection Manager is specifically designed to take full advantage of the enhanced e*Gate functionality. In e*Gate 4.5.1 or earlier, this enhanced functionality is visible, but ignored.

The Connection Manager is controlled as described in [Chapter 6](#). If you choose to control the e*Way Connections manually, you may find the following chart helpful.

Table 12 e*Way Connection Controls

Method	Automatic	On-Demand	Manual
onConnectionUp	yes	no	no
onConnectionDown	yes	yes—but only if the connection attempt fails	no
Automatic Transaction (XA)	yes	no	no
Manual Transaction (XA)	yes	no	no
connect	no	no	yes
isConnect	no	no	yes
disconnect	no	no	yes
timeout or connect	yes	yes	no
verify connection interval	yes	no	no

Controlling When a Connection is Made

By using Connector Settings, you may have e*Way connections controlled manually (through the Collaboration) or automatically (through the e*Way Connection Configuration). If you choose to control the connection manually, you can have the e*Way Connection made:

- When the Collaboration is loaded
- When the Collaboration is executed
- By using an additional connection method in the ETD
- By overriding any custom values you have assigned in the Collaboration

- By using the `isConnected()` method (if your ETD has multiple connections, the `isConnected()` method is called *per connection*)

Controlling When a Connection is Disconnected

In addition to controlling when a connection is made, you can also control when an e*Way Connection is terminated or disconnected either manually or automatically. You can have the e*Way Connection disconnect:

- At the end of a Collaboration
- At the end of the execution of the Collaborations Business Rules
- During a timeout
- After a method call

Controlling the Connectivity Status

You can control how often the e*Way Connection checks to verify is still live; see:

[Connector Settings](#) on page 122 (for Oracle DBMS)

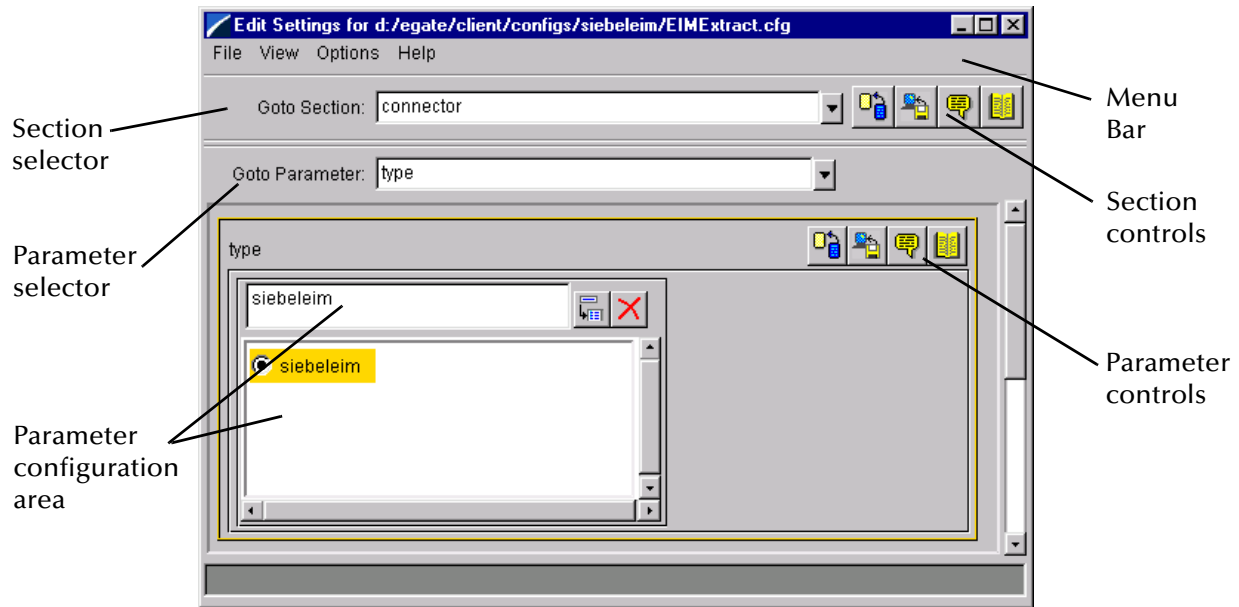
[Connector Settings](#) on page 128 (for DB2 UDB)

[Connector Settings](#) on page 133 (for SQL Server)

4.5 Using the e*Way Configuration Editor

The e*Way's default configuration parameters are stored in an ASCII text file with a **.def** extension. The e*Way Editor provides a simple graphical interface for viewing and changing those parameters to create a working configuration file pair (**.cfg** and **.sc**). The editor has the same general appearance for either e*Way or e*Way Connection configuration (as shown in Figure 41).

Figure 41 The e*Way Configuration Editor







The e*Way Configuration Editor controls fall into one of six categories:

- The **Menu bar** allows access to basic operations (e.g., saving the configuration file, viewing a summary of all parameter settings, and launching the Help system)
- The **Section selector** at the top of the Editor window enables you to select the category of the parameters you wish to edit
- **Section controls** enable you to restore the default settings, restore the last saved settings, display tips, or enter comments for the currently selected section
- The **Parameter selector** allows you to jump to a specific parameter within the section, rather than scrolling
- **Parameter controls** enable you to restore the default settings, restore the last saved settings, display tips, or enter comments for the currently selected parameter
- **Parameter configuration controls** enable you to set the e*Way's various operating parameters

4.5.1 Section and Parameter Controls

The section and parameter controls are shown in Table 13 below.

Table 13 Parameter and Section Controls

Button	Name	Function
	Restore Default	Restores default values
	Restore Value	Restores saved values
	Tips	Displays tips
	User Notes	Enters user notes



Note: The section controls affect all parameters in the selected section, whereas the parameter controls affect only the selected parameter.

4.5.2 Parameter Configuration Controls

Parameter configuration controls fall into one of two categories:

- Option buttons
- Selection lists, which have controls as described in Table 14

Table 14 Selection List Controls

Button	Name	Function
	Add to List	Adds the value in the text box to the list of available values.
	Delete Items	Displays a “delete items” dialog box, used to delete items from the list.

4.5.3 File Menu Selections

- **Save**
 - ♦ If you are editing a **.def** file, selecting **Save** creates a working configuration file pair (**.cfg** and **.sc**) from the file you are editing.
 - ♦ If you are editing an existing **.cfg** file, selecting **Save** overwrites the existing configuration file pair (**.cfg** and **.sc**).
- **Promote to Run Time**
 - ♦ Selecting **Promote to Run Time** promotes the configuration file pair (**.cfg** and **.sc**) from the Sandbox to the registry, committing them to the current schema.

For more information, see the *Getting Started* chapter of the *e*Gate Integrator User's Guide*.

4.5.4 Command-line Configuration

In the **Additional Command Line Arguments** box, type any additional command line arguments that the e*Way may require, taking care to insert them *at the end* of the existing command-line string. Be careful not to change any of the default arguments unless you have a specific need to do so.

4.5.5 Getting Help

To launch the e*Way Editor's Help system

From the **Help** menu, select **Help topics**.

To display tips regarding the general operation of the e*Way

From the **File** menu, select **Tips**.

To display tips regarding the selected Configuration Section

In the **Section Control** group, click .

To display tips regarding the selected Configuration Parameter

In the **Parameter Control** group, click .

Note: *Tips are displayed and managed separately from the online Help system. You cannot search for Tips within the Help system, or view Help system topics by requesting Tips.*

For detailed descriptions and procedures for using the e*Way Configuration Editor, see the *Working with e*Ways* chapter of the *e*Gate Integrator User's Guide*.

4.6 Troubleshooting the e*Way

In the initial stages of developing your e*Gate Integrator system administration system, most problems with e*Ways can be traced to configuration.

4.6.1 Configuration Problems

In the Schema Designer

- Does the e*Way have the correct Collaborations assigned?
- Do those Collaborations use the correct Collaboration Services?
- Is the logic correct within any Collaboration Rules script employed by this e*Way's Collaborations?
- Do those Collaborations subscribe to and publish Events appropriately?
- Are all the components that provide information to this e*Way properly configured, and are they sending the appropriate Events correctly?
- Are all the components to which this e*Way sends information properly configured, and are they subscribing to the appropriate Events correctly?

In the e*Way Editor

- Check that all e*Way connection options are set appropriately.
- Check that all settings you changed are set correctly.
- Check all required changes to ensure they have not been overlooked.
- Check the defaults to ensure they are acceptable for your installation.

On the e*Way's Participating Host

- Check that the Participating Host is operating properly, and that it has sufficient disk space to hold the IQ data that this e*Way's Collaborations publish.
- Check that the *path* environmental variable includes the location of the Siebel EIM dynamically-loaded libraries. The name of this variable on the different operating systems is:
 - ♦ PATH (Windows)
 - ♦ LD_LIBRARY_PATH (Solaris)
 - ♦ LIBPATH (AIX)

In the External Application

- Check that the application is configured correctly, is operating properly, and is sending or receiving the correct data appropriately.

4.6.2 System-related Problems

- Check that the connection between the external application and the e*Way is functioning appropriately.
- Once the e*Way is up and running properly, operational problems can be due to:
 - ♦ External influences (network or other connectivity problems).
 - ♦ Problems in the operating environment (low disk space or system errors)
 - ♦ Problems or changes in the data the e*Way is processing.
 - ♦ Corrections required to Collaboration Rules scripts that become evident in the course of normal operations.

One of the most important tools in the troubleshooter's arsenal is the e*Way log file. See the *e*Gate Integrator Alert and Log File Reference Guide* for an extensive explanation of log files, debugging options, and using the e*Gate Schema Manager system to monitor operations and performance.

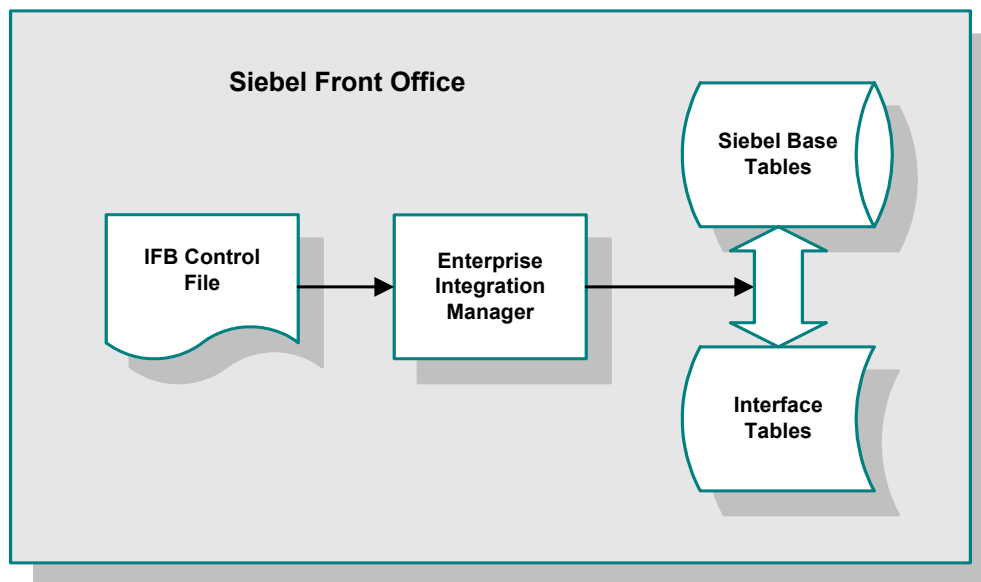
Operational Overview

This chapter provides an overview of the basic functionality of the SeeBeyond Java e*Way Intelligent Adapter for Siebel EIM.

5.1 Siebel EIM

The Siebel Enterprise Integration Manager is the first step in exporting, and the last step in importing, data. The database consists of two main groups of tables, interface tables and base tables. The Siebel client application communicates directly with the highly normalized base tables. The interface tables are used as a staging area for importing, exporting, deleting and merging logical groups of data. Each interface table represents a subset of the data in a specific base table.

Figure 42 Siebel Enterprise Integration Manager



When run, EIM coordinates the transfer of data between the base and interface tables. In addition, EIM creates and writes codes and row IDs to the base tables that directly correspond to the complex set of business rules used by the Siebel client application to display data. The EIM process must run on the server also running the Transaction Processor.

Typically, files to be imported have a predecessor-successor relationship, because Siebel requires files to be loaded in a certain manner. Thus any file depends on the completion of the preceding file. For example, if a project were receiving accounts, contacts and opportunities from an external source, they would receive either of the following:

- Three files from the external source
- One large file containing all of the pertinent information within one record

In scenario one, the three files would be Accounts, Contacts, and Opportunities, and Siebel requires the files be loaded in that order. Once EIM has completed interfacing Accounts, and the Accounts are loaded into the base tables, Contacts are loaded and an association is linked between the two entities. Opportunities follow the same logic.

In scenario two, the file is translated within e*Gate to appear as if three files were loaded into the correct interface tables. Both Scenario one and scenario two have the same predecessor-successor relationship that is inherent in Siebel. To load contacts successfully, the account that the contact belongs to must be loaded first.

See the *Siebel Administration Guide* for more details regarding Siebel EIM.

5.2 IFB Configuration File

A Siebel configuration file (*.ifb) is used to determine what data types are loaded and how. The control file follows a certain format—it tells the interface how to log into the database and what process to run. It also lists the columns that the interface does *not* populate (in Siebel-inbound mode), thus preventing erroneous error messages. The IFB control file can be generated automatically by the e*Way or created manually, as configured by the user.

Following is an example of the IFB file format:

```
[Siebel Interface Manager]
  PASSWORD = "sadmin"
  PROCESS = Import Accounts
  TABLEOWNER = "siebel"
  USER NAME = "sadmin"

[Import Accounts]
  TYPE = SHELL
  INCLUDE = "Import Accounts without Addresses w/o Position"

[Import Accounts without Addresses w/o Position]
  TYPE = IMPORT
  BATCH = 1201
  ONLY BASE TABLES = S_PARTY, S_ORG_EXT, S_ADDR_ORG
  TABLE = EIM_ACCOUNT
  USING SYNONYMS = FALSE
  DEFAULT COLUMN = ACTIVE_FLG, "Y"
  DEFAULT COLUMN = DISA_CLEANSE_FLG, "N"
  DEFAULT COLUMN = EVT_LOC_FLG, "N"
  DEFAULT COLUMN = FCST_ORG_FLG, "N"
  DEFAULT COLUMN = INT_ORG_FLG, "Y"
  DEFAULT COLUMN = PROSPECT_FLG, "N"
  DEFAULT COLUMN = PRTNR_FLG, "N"
  DEFAULT COLUMN = PRTNR_PUBLISH_FLG, "N"
  DEFAULT COLUMN = RPLCD_WTH_CMPT_FLG, "N"
  DEFAULT COLUMN = ROOT_PARTY_FLG, "N"
  DEFAULT COLUMN = ADDR_ACTIVE_FLG, "Y"
  DEFAULT COLUMN = ADDR_BL_ADDR_FLG, "Y"
  DEFAULT COLUMN = ADDR_DISACLEANSEFL, "N"
  DEFAULT COLUMN = ADDR_MAIN_ADDR_FLG, "Y"
  DEFAULT COLUMN = ADDR_NAME_LOCK_FLG, "N"
  DEFAULT COLUMN = ADDR_SHIP_ADDR_FLG, "Y"
  INSERT ROWS = TRUE
  UPDATE ROWS = TRUE
```

The **IFB Filename** indicated in the configuration file for the e*Way is used to process the data.

5.3 Siebel EIM e*Way Operation

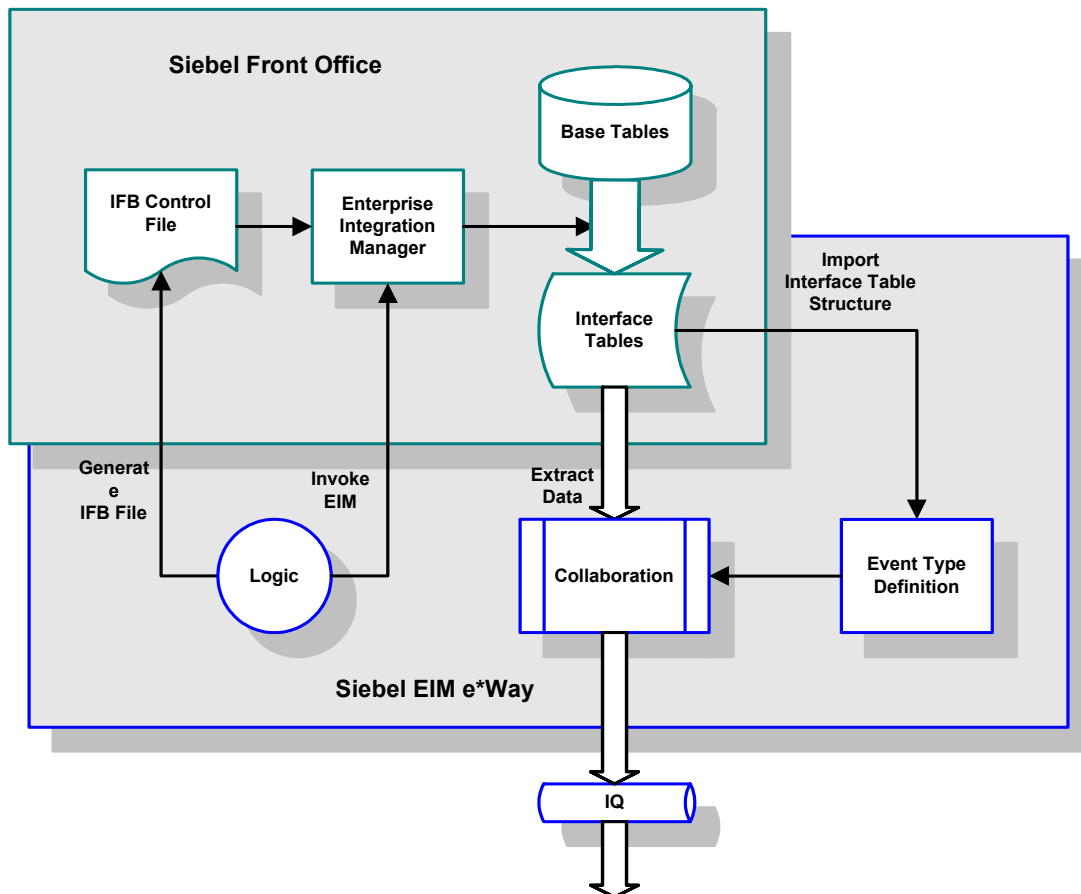
The e*Way uses Java methods to exchange data with the external system, package data as e*Gate *Events*, send those Events to Collaborations, and manage the connection between the e*Way and the external system. These topics are covered in detail in the *e*Gate Integrator User's Guide*.

The Siebel-to-e*Gate operation is dealt with first, since it is less complex than the e*Gate-to-Siebel case.

5.3.1 Siebel to e*Gate

Overview

Figure 43 Siebel-to-e*Gate Process Flow



- 1 Following a prescribed schedule, the Siebel EIM e*Way either generates or retrieves an IFB Control File, and sends it to Siebel.
- 2 The e*Way then invokes the Siebel EIM process.

- 3 Siebel EIM then copies data from the Siebel Base Tables into the Interface Tables, as prescribed by the IFB Control File.
- 4 The Siebel EIM e*Way extracts the data from the interface tables and maps it into the correct Event Type Definition, following a pre-defined Collaboration.
- 5 The data is then passed to other e*Gate components for further processing (if required) and routing to the target application.

Export Procedure

Following are the logical steps for exporting data into Siebel, using the provided Java methods:

Note: *If the IFB file already exists, there is no need to generate it.*

- 1 Generate the IFB if it does not already exist (using **generateIfb()**).
- 2 Start the Siebel task (using **startTask()**).
- 3 Perform other operations if necessary.

Note: *If no other operations need to be performed, steps 2 and 4 can be combined by using **startTaskAndWaitUntilDone()**.*

- 4 Wait until the Siebel task is done (using **waitUntilDone(taskNum)**).
- 5 Verify the export task is successful.
- 6 Retrieve records from the interface table.
- 7 Clean up the exported records in the interface table if the operation is successful.

Note: *You must perform the steps to **set** and **get** the last export time (using **getLastExportTime()** and **setLastExportTime(export_time)**) if you do not want to retrieve previously-exported records.*

Example

The sample schema **EIM_Extract** on page 52 provides an example of exporting data from the Siebel EIM_ACCOUNT interface table. This sample schema follows the procedure shown below:

- 1 Retrieve the **timestamp** of the last export.
- 2 Generate the IFB file.
- 3 Start the Siebel server task and wait until it is done.

If the export operation is successful,

- 4 Save the export time for the next export operation.
- 5 Retrieve the records from the interface table EIM_ACCOUNT.
- 6 Clean up the exported records in the interface table for this batch operation

Export Code

```
public boolean executeBusinessRules() throws java.lang.Exception
{
    boolean retBoolean = true;
    int ind;
    String payload = null;
    String lasttts;
    String currentStr;
    java.sql.Timestamp currenttts = new java.sql.Timestamp(0);
    String filename = null;
    String retVal = null;

    EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Input trigger data: " + getIn2().getPayload() );

    EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Write values into IFB file...");

    // Generate IFB file

    getInOut().getIFB_Control().getIFB_Content().getHeaderName().setV
alue("Siebel Interface Manager");

    getInOut().getIFB_Control().getIFB_Content().getHeader().getPROCE
SS().setValue("Export Accounts");

    getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getProcessName().setValue("Export Accounts");

    getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getIFBProcess().getTYPE().setValue("EXPORT");

    getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getIFBProcess().getBATCH().setValue(getInOut().getIFB_Control().g
etBatchNumber());

    getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getIFBProcess().getTable(0).setValue("EIM_ACCOUNT");

    getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getExportProcess().getCLEAR_INTERFACE_TABLE().setValue("TRUE");

    getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getExportProcess().getEXPORT_ALL_ROWS().setValue("FALSE");

    // Retrieve the current database timestamp
    // For Oracle, we need to specify the timestamp format that
we want to retrieve
    // The query to retrieve the timestamp is also database
specific.
    // Please see documentation for the correct query to use.
    com.stc.eways.jdbcx.SqlStatementAgent alterAgent =
getIn1().getSqlAgent("alterAgent");
    alterAgent.execute("alter session set NLS_DATE_FORMAT =
'YYYY-MM-DD HH24:MI:SS'");
    com.stc.eways.jdbcx.SqlStatementAgent selectAgent =
getIn1().getSqlAgent("selectAgent");
    com.stc.eways.jdbcx.ResultSetAgent rs =
selectAgent.executeQuery("select sysdate from dual");
    while (rs.next())
    {
        currenttts = rs.getTimestamp(1);
    }
}
```

```

        // Retrieve the last export time to exclude previously
        exported records in this export operation
        lasttts = getInOut().getIFB_Control().getLastExportTime();
        ind = currenttts.toString().indexOf(".");
        currentStr = currenttts.toString().substring(0, ind);

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getExportProcess().getEXPORT_MATCHES().setValue("S_ORG_EXT,
(LAST_UPD >= ' " + lasttts + "' AND LAST_UPD < ' " + currentStr +
"'"));

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Current Time: " + currentStr);

        // Generate the IFB file
        filename = getInOut().getIFB_Control().generateIfb();

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Creating IFB file: " + filename);
        retVal =
getInOut().getEIM_Control().startTaskAndWaitUntilDone();

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Ran EIM returned: " + retVal);
        if (retVal.equals("EIM_SUCCESS"))
        {
            // Save the export time
            getInOut().getIFB_Control().setLastExportTime(currentStr);

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Set current time to file: " + currentStr);
        }
        else
        {
        }

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Ran EIM returned status: "
+getInOut().getEIM_Control().getServerTasks().getRow().getTK_STAT
US() + " state: " +
getInOut().getEIM_Control().getServerTasks().getRow().getTK_DISP_
RUNSTATE());

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Creating result set....");

        // Retrieve records from EIM_ACCOUNT interface table and
        publish it to the queue
        getIn1().getEIM_ACCOUNT().select("addr_addr_name is not
null");
        while (getIn1().getEIM_ACCOUNT().next())
        {

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Has data...." + getIn1().getEIM_ACCOUNT().getName());
            payload = getIn1().getEIM_ACCOUNT().getName() + "|" +
getIn1().getEIM_ACCOUNT().getLOC();
            getOut1().setPayload(payload);
            getOut1().send();
        }

        // Clean up the exported records

```

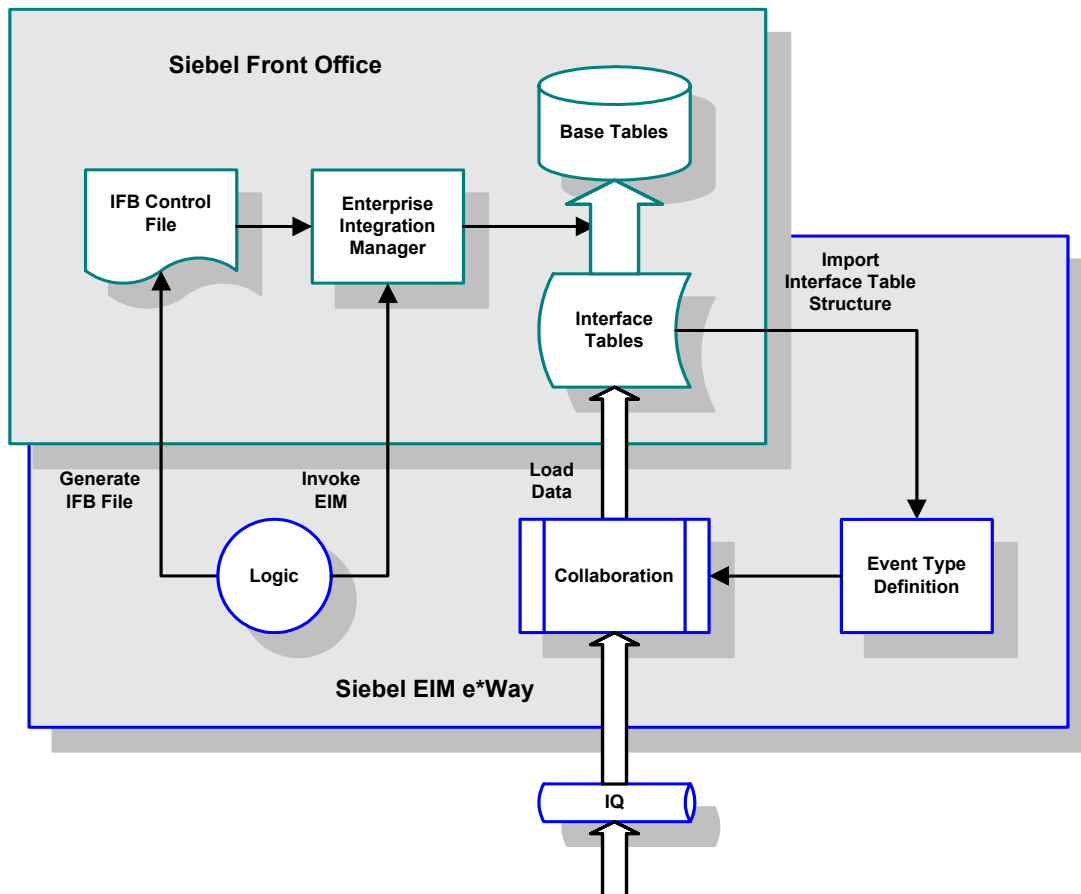
```
        com.stc.eways.jdbcx.SqlStatementAgent deleteAgent =
getIn1().getSqlAgent("deleteAgent");
        deleteAgent.executeUpdate("delete from eim_account where
IF_ROW_STAT = 'EXPORTED' and IF_ROW_BATCH_NUM = '" +
getInOut().getIFB_Control().getBatchNumber() + "'");

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>>> Delete from eim_account ...");
        return retBoolean;
    }
}
```


5.3.2 e*Gate to Siebel

Overview

Figure 44 e*Gate-to-Siebel Process Flow



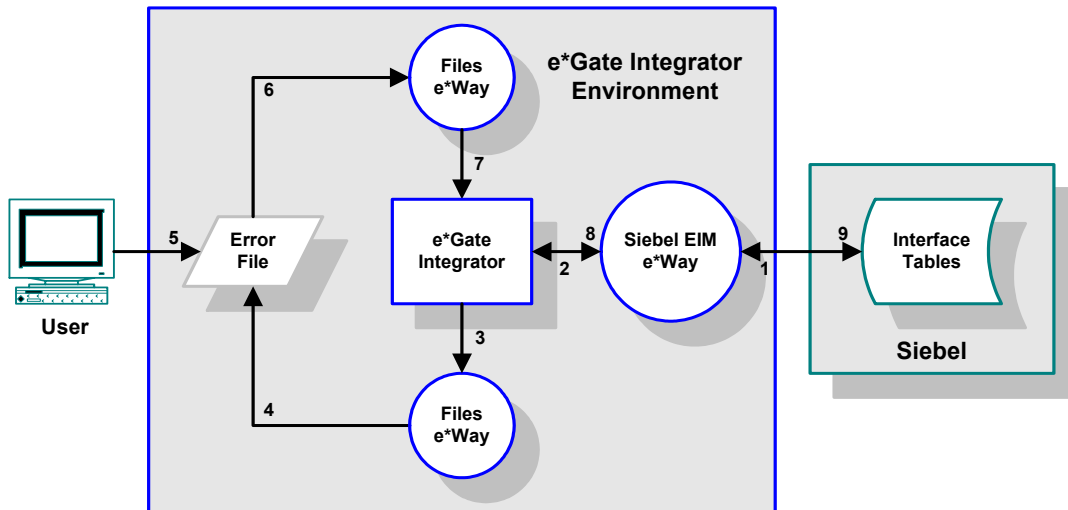
- 1 The Siebel EIM e*Way extracts data from an Intelligent Queue for processing.
- 2 The e*Way processes the information following a Collaboration previously created using structural information extracted from Siebel, and inserts validated rows into the Siebel Interface Tables.

Note: *e*Gate allows the population of ROW_ID, IF_ROW_BATCH_NUM, and IF_ROW_STAT, according to the chosen naming standards during the insert statement.*

- 3 An IFB control file is built (or retrieved) and created to the Siebel system for use by the EIM in populating the Base Tables with data from the Interface Tables.
- 4 The e*Way initiates the Siebel Enterprise Integration Manager (EIM) to load the data from the Interface Tables into the appropriate Siebel Base Tables.

- 5 Upon completion of the EIM process, e*Gate interrogates the `IF_ROW_STAT` column in the Siebel Interface Tables to verify the correct transfer of data to the Base Tables. All records transferred successfully are deleted from the Interface Tables.

Figure 45 e*Gate-to-Siebel Error Processing



- 6 If any errors have occurred, the rejected records are pulled back into e*Gate by the EIM e*Way, placed in a error log file, and deleted from the Interface Tables.
- 7 User intervention is required to analyze and correct the data, and submit the records for reprocessing.

Import Procedure

Following are the logical steps for importing data into Siebel, using the provided Java methods:

- 1 Generate the IFB, if it does not already exist (using `generateIfb()`).
- 2 Populate the interface table.
- 3 Start the Siebel task (using `startTask()`).
- 4 Perform other operations if necessary.

Note: *If no other operations need to be performed, steps 3 and 5 can be combined by using `startTaskAndWaitUntilDone()`.*

- 5 Wait until the Siebel task is done (using `waitUntilDone(taskNum)`).
- 6 Verify the import task is successful.
- 7 Clean up the imported records in the interface table if the operation is successful.
- 8 Check and perform necessary operations if there is any record fail to import.
- 9 You also can perform additional steps to `set` and `get` the state of the last import operation (using `getImportState()` and `setImportState(MSG_ACKED)`).

Note: If the IFB file already exists, there is no need to generate it.

Example

The sample schema [EIM_Post](#) on page 53 provides an example of importing data into the Siebel EIM_ACCOUNT interface table. This sample schema follows the procedure shown below:

Note: Performance can be improved if batch mode operation is used.

- 1 Verify the state of the last import operation.
- 2 Log the import state if it is not acknowledged and continue the operation.
- 3 Populate the interface table EIM_ACCOUNT.
- 4 Update the import state file to **inserted**.
- 5 Generate the IFB file.
- 6 Start the Siebel server task and wait until it is done.
- 7 Set the import state to **processed**.

If the import operation is successful,

- 8 Clean up the imported records in the interface table for this batch operation.
- 9 Check if any records failed to import.
- 10 Update the import state file to **acknowledged**.

Import Code

```
public boolean executeBusinessRules() throws java.lang.Exception
{
    boolean retBoolean = true;
    String MSG_ACKED = "0";
    String MSG_INSERTED = "1";
    String MSG_PROCESSED = "2";
    String filename = null;
    String taskState = "START";
    String taskNum = null;
    String importState = null;

    // Verify the state of the last import operation
    // Log the import state if it is not acknowledged and continue
the operation
    importState = getInOut().getIFB_Control().getImportState();
    if (importState.equals(MSG_ACKED))
    {
        EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Import state: " + importState + ". Message
acknowledged in the last import.");
    }
    else
    {
        EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Import state is not MSG_ACKED, it's : " + importState
```

```
+ ". The program will continue to import the records in the
interface table after resetting the import state.");
    getInOut().getIFB_Control().setImportState(MSG_ACKED);
}

// Populate the interface table EIM_ACCOUNT
// Note that this example insert the record one at a time.
// Users can significantly increase the performance by
inserting the records in batch mode.
for( int ii=0;ii < getIn1().countDataContract();ii++)
{

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>>> Creating updatable result set...");
    getOut1().setScrollTypeToScrollSensitive();
    getOut1().setConcurrencyToUpdatable();
    getOut1().getEIM_ACCOUNT().select("");
    getOut1().getEIM_ACCOUNT().moveToInsertRow();

getOut1().getEIM_ACCOUNT().setROW_ID(Long.toString(System.current
TimeMillis()));

getOut1().getEIM_ACCOUNT().setIF_ROW_BATCH_NUM(getInOut().getIFB_
Control().getBatchNumber());
    getOut1().getEIM_ACCOUNT().setIF_ROW_STAT("FOR_IMPORT");
    getOut1().getEIM_ACCOUNT().setPARTY_TYPE_CD("SEEBEYOND
CORP_PARTY");
    getOut1().getEIM_ACCOUNT().setADDR_ADDR_NAME("Address name
" + Integer.toString(ii+1));

getOut1().getEIM_ACCOUNT().setPARTY_UID(getIn1().getDataContract(
ii).getCustomerNumber());

getOut1().getEIM_ACCOUNT().setName(getIn1().getDataContract(ii).ge
tAccountName());

getOut1().getEIM_ACCOUNT().setLOC(getIn1().getDataContract(ii).ge
tSiteLocation());

getOut1().getEIM_ACCOUNT().setADDR_FAX_PH_NUM(getIn1().getDataCon
tract(ii).getMainFax());

getOut1().getEIM_ACCOUNT().setADDR_PH_NUM(getIn1().getDataContra
ct(ii).getMainPhone());

getOut1().getEIM_ACCOUNT().setADDR_ADDR(getIn1().getDataContract(
ii).getStreet());

getOut1().getEIM_ACCOUNT().setADDR_CITY(getIn1().getDataContract(
ii).getCity());

getOut1().getEIM_ACCOUNT().setADDR_STATE(getIn1().getDataContract
(ii).getState());

getOut1().getEIM_ACCOUNT().setADDR_ZIPCODE(getIn1().getDataContra
ct(ii).getPostalCode());

getOut1().getEIM_ACCOUNT().setADDR_COUNTRY(getIn1().getDataContra
ct(ii).getCountry());

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>>> Inserting row...");
    getOut1().getEIM_ACCOUNT().insertRow();
}
```

```
// users will need to commit the record otherwise Siebel will
not see it
    getOut1().commit();

    // update the import state file to "inserted"
    getInOut().getIFB_Control().setImportState(MSG_INSERTED);
    importState = MSG_INSERTED;

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Interface table records committed. Import state set
to MSG_INSERTED.");

    // Prepare to generate the IFB file; this is an optional step
if the IFB file already exists
    if (importState.equals(MSG_INSERTED))
    {

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Records committed, will set values for IFB file ...");

getInOut().getIFB_Control().getIFB_Content().getHeaderName().setV
alue("Siebel Interface Manager");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getProcessName().setValue("Import Accounts");

getInOut().getIFB_Control().getIFB_Content().getHeader().getPROCE
SS().setValue("Import Accounts");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getIFBProcess().getTYPE().setValue("SHELL");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(0).
getIFBProcess().getINCLUDE(0).setValue("Import Accounts without
Addresses w/o Position");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).
getProcessName().setValue("Import Accounts without Addresses w/o
Position");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).
getIFBProcess().getTYPE().setValue("IMPORT");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).
getIFBProcess().getBATCH().setValue(getInOut().getIFB_Control().g
etBatchNumber());

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).
getIFBProcess().getONLY_BASE_TABLES().setValue("S_PARTY,
S_ORG_EXT, S_ADDR_ORG");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).
getIFBProcess().getTABLE(0).setValue("EIM_ACCOUNT");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).
getIFBProcess().getUSING_SYNONYMS().setValue("FALSE");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).
getImportProcess().getINSERT_ROWS().setValue("TRUE");

getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).
getImportProcess().getUPDATE_ROWS().setValue("TRUE");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(0).setValue("ACTIVE_FLG,  
\ "Y\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(1).setValue("DISA_CLEANSE_FLG,  
\ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(2).setValue("EVT_LOC_FLG,  
\ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(3).setValue("FCST_ORG_FLG,  
\ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(4).setValue("INT_ORG_FLG,  
\ "Y\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(5).setValue("PROSPECT_FLG,  
\ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(6).setValue("PRTNR_FLG,  
\ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(7).setValue("PRTNR_PUBLISH_F  
LG, \ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(8).setValue("RPLCD_WTH_CMPT_  
FLG, \ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(9).setValue("ROOT_PARTY_FLG,  
\ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(10).setValue("ADDR_ACTIVE_FL  
G, \ "Y\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(11).setValue("ADDR_BL_ADDR_F  
LG, \ "Y\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(12).setValue("ADDR_DISACLEAN  
SEFL, \ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(13).setValue("ADDR_MAIN_ADDR  
_FLG, \ "Y\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(14).setValue("ADDR_NAME_LOCK  
_FLG, \ "N\ " ");
```

```
getInOut().getIFB_Control().getIFB_Content().getIFB_Iteration(1).  
getImportProcess().getDEFAULT_COLUMN(15).setValue("ADDR_SHIP_ADDR  
_FLG, \ "Y\ " ");
```

```

        filename = getInOut().getIFB_Control().generateIfb();

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Creating IFB file: " + filename);

        // Start the Siebel server task and wait until it is done
        taskNum=getInOut().getEIM_Control().startTask();

taskState=getInOut().getEIM_Control().waitUntilDone(taskNum);

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Ran EIM returned task num: " + taskNum + " and
returned: " + taskState);

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Ran EIM returned status: "
+getInOut().getEIM_Control().getServerTasks().getRow().getTK_STAT
US() + " state: " +
getInOut().getEIM_Control().getServerTasks().getRow().getTK_DISP_
RUNSTATE());

        // Set the import state to "processed"
getInOut().getIFB_Control().setImportState(MSG_PROCESSED);
importState = MSG_PROCESSED;

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Ran EIM. Import state set to MSG_PROCESSED.");
    }
    else
    {

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Skipped running of EIM. Import state is not
MSG_INSERTED, it's : " + importState);
    }

        // If the import operation is successful, clean up the
imported records in the interface table for this batch operation
        // Check if there is any record fail to import.
        if (importState.equals(MSG_PROCESSED) &&
taskState.equals("EIM_SUCCESS"))
        {
            com.stc.eways.jdbcx.SqlStatementAgent deleteAgent =
getOut1().getSqlAgent("deleteAgent");
            deleteAgent.executeUpdate("delete from eim_account where
IF_ROW_STAT = 'IMPORTED' and IF_ROW_BATCH_NUM = '" +
getInOut().getIFB_Control().getBatchNumber() + "'");
            getOut1().commit();

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Delete from eim_account ...");
            com.stc.eways.jdbcx.SqlStatementAgent notImportAgent =
getOut1().getSqlAgent("notImportAgent");
            com.stc.eways.jdbcx.ResultSetAgent rs =
notImportAgent.executeQuery("select count(1) from eim_account
where IF_ROW_STAT != 'IMPORTED' and IF_ROW_BATCH_NUM = '" +
getInOut().getIFB_Control().getBatchNumber() + "'");
            while (rs.next())
            {

EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA
TION,">>>> Number of NOT IMPORTED records: " + rs.getString(1));
            }
            getInOut().getIFB_Control().setImportState(MSG_ACKED);

```

```
EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA  
TION,">>>>> Cleaned up interface table. Import state set to  
MSG_ACKED.");  
    }  
    else  
    {  
  
EGate.traceln(EGate.TRACE_COLLABSERVICE,EGate.TRACE_EVENT_INFORMA  
TION,">>>>> Did not clean up interface table. Import state: " +  
importState);  
    }  
    return retBoolean;  
}
```


5.4 Event Type Definitions

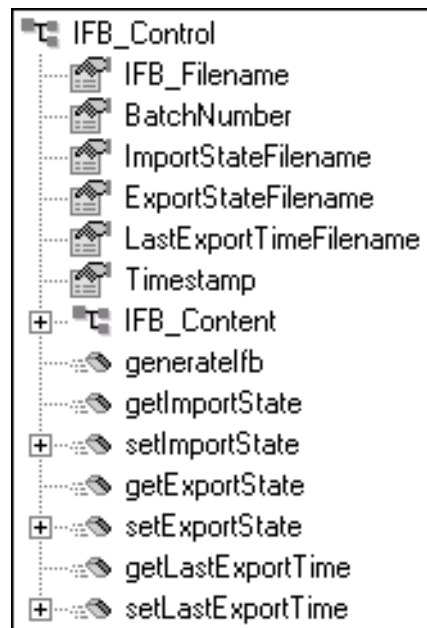
5.4.1 SiebelEIM

The Siebel EIM e*Way ETD, **SiebelEIM.xsc**, provides the main user interface for the e*Way. It contains two main nodes: **IFB_Control** and **EIM_Control**.

IFB_Control

If you want to have the e*Way generate the Siebel IFB file automatically, the attribute nodes in **IFB_Control** (shown in Figure 46) take their values from the e*Way's configuration parameters (see **Siebel Control** on page 116). Otherwise, you must populate them manually.

Figure 46 IFB_Control



Note: *If you redefine any of the properties in this ETD, the values you place in the ETD will take precedence over the values obtained from the configuration file.*

Of particular interest is the **Timestamp** node, which is used if you want to obtain records from the database within a certain time frame. The procedure for using this node is described in **Using the Timestamp Node** on page 46.

Note: *Timestamps are obtained from the Siebel Database Server. If different Siebel components are installed on different systems (within the same overall installation), they must be synchronized for this feature to work properly.*

The **IFB_Control** node also contains a secondary node, **IFB_Content**, and seven method nodes:

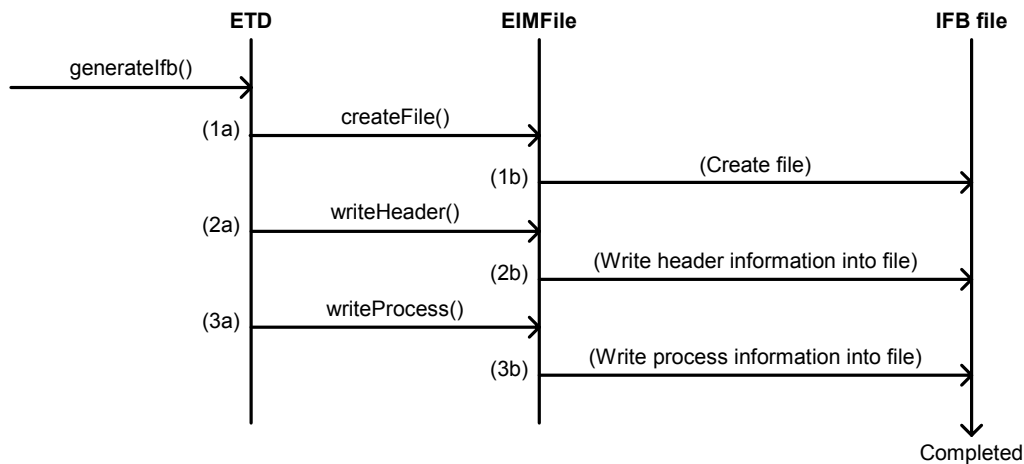
generatelfb	setExportState
getImportState	getLastExportTime
setImportState	setLastExportTime
getExportState	

These methods are described below and in **IFB_Control Class** on page 141. The **IFB_Content** node is described in the following section.

generatelfb()

This is the method to call if you want to have the e*Way generate the IFB file automatically. In this case, the e*Way takes the content of the input data and creates the Siebel IFB file at the location specified in the configuration file.

*Note: e*Gate must have a network path to the generated IFB file, such as a NFS mounted directory, UNC or local directory, or the e*Way must run on the same host as the Siebel server. No FTP capability is provided.*

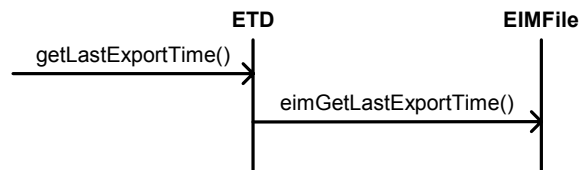


As shown in the preceding diagram, calling **generateIfb()** initiates the following sequence of events:

- 1 Calls **createFile()**, which creates the file (if the file already exists, it will be overwritten).
- 2 Calls **writeHeader()**, which writes the Header section into the IFB file.
- 3 Calls **writeProcess()**, which writes the Process section into the IFB file, completing the file.

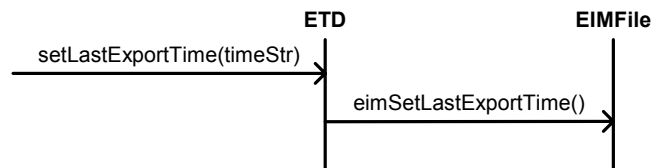
getLastExportTime()

This method calls **eimGetLastExportTime()** to get the time that data was last exported from the file that is specified in the e*Way configuration file, or the ETD.



setLastExportTime(timeStr)

This method calls **eimSetLastExportTime()** to set the time that data was last exported into the file that is specified in the e*Way configuration file, or the ETD.



IFB_Content

The **IFB_Content** node is used to hold information that creates the IFB file and its parameters. It contains three tertiary nodes, **Header_Name**, **Header**, and the repeating node **IFB_Iteration**. It also contains one method, **countIFB_Iteration**.

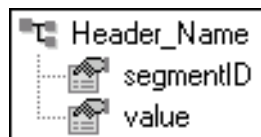
Figure 47 IFB_Content



Header_Name

The **Header_Name** node (see Figure 48) contains two attributes nodes, which specify the header name of the IFB File.

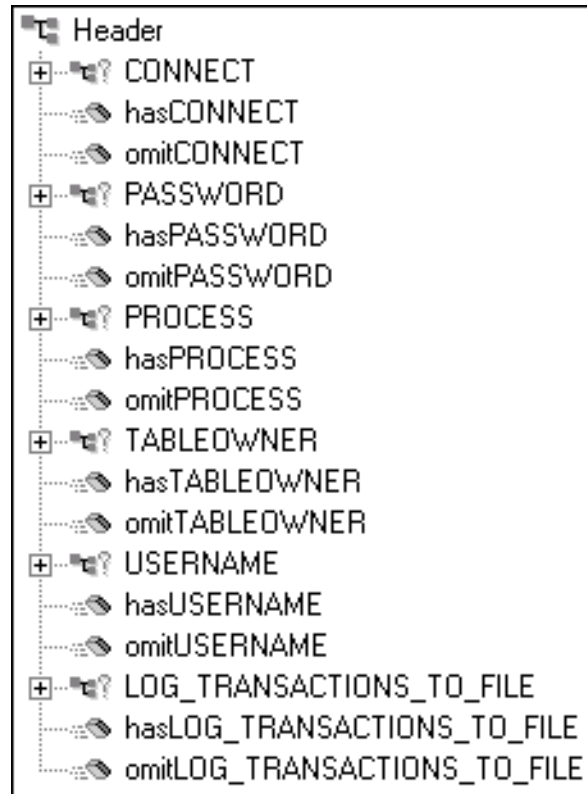
Figure 48 Header_Name



Header

The **Header** node (see Figure 49) contains attribute nodes for user and table information appearing in the IFB File header (compare with the example file shown in [IFB Configuration File](#) on page 75).

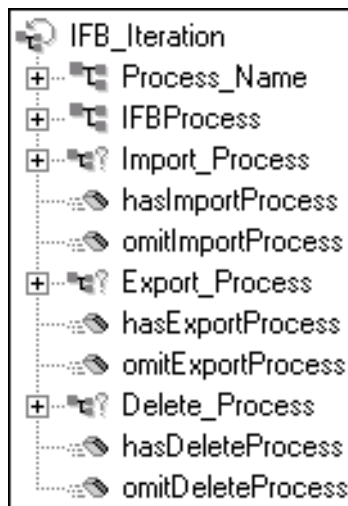
Figure 49 Header



IFB_Iteration

The **IFB_Iteration** node (see Figure 50) contains attribute nodes for controlling the IFB process.

Figure 50 IFB_Iteration



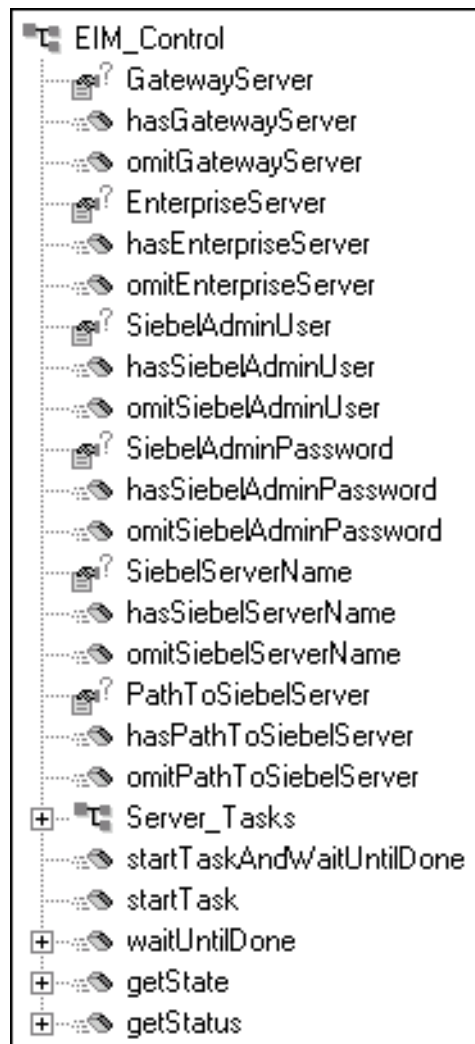
EIM_Control

EIM_Control, shown in Figure 51, contains attribute nodes for the Siebel server information. These are taken from the e*Way configuration file when automatically generating the IFB file (see [Siebel Server](#) on page 113). It also contains a tertiary node **Server_Tasks** containing the output of the server tasks, and the method nodes:

startTaskAndWaitUntilDone	getState
startTask	getStatus
waitUntilDone	

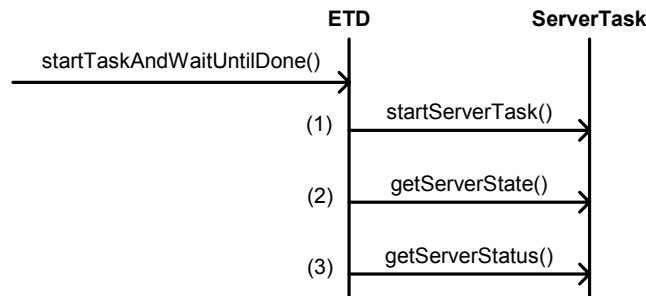
These methods are described below and in [EIM_Control Class](#) on page 137.

Figure 51 EIM_Control



startTaskAndWaitUntilDone()

This method obtains the task number of the job associated with the IFB file, and monitors the job by checking its state. If the job exits with an error, it also checks the error status.

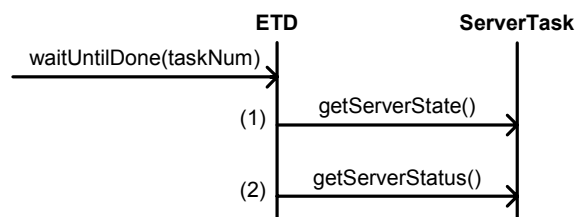


As shown above, calling **startTaskAndWaitUntilDone()** initiates the following sequence of events:

- 1 Calls **startServerTask()**, which issues the command:
`start task for component EIM server servername with config=ifb_file`
 and obtains the task number.
- 2 Calls **getServerState()**, which issues the command:
`list task tasknum for servername`
 and obtains the state of the job. It then waits 10 seconds and repeats until the job is completed or fails.
- 3 If the job fails, it calls **getServerStatus()** and obtains the status of the job.

waitUntilDone(taskNum)

This method monitors the job by checking its state. If the job exited with error, it will also check the error status.



As shown above, calling **waitUntilDone(taskNum)** initiates the following sequence of events:

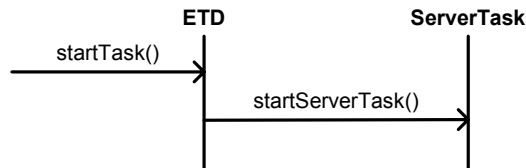
- 1 Calls **getServerState()**, which issues the command:
`list task tasknum for servername`
 and obtains the state of the job. It then waits 10 seconds and repeats until the job is completed or fails.

- 2 If the job fails, calls **getServerStatus()** and obtains the status of the job.

startTask()

This method calls **startServerTask()**, which issues the command

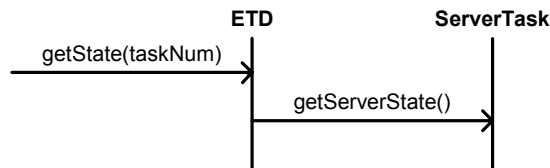
```
start task for component EIM server servername with config=ifb_file  
and obtains task number associated with the IFB file.
```



getState(taskNum)

This method calls **getServerState()**, which issues the command:

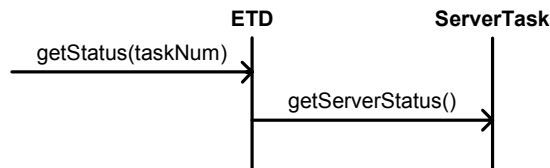
```
list task tasknum for servername  
and obtains the state of the job.
```



getStatus(taskNum)

This method calls **getServerStatus()**, which issues the command:

```
list task tasknum for servername  
and obtains the status of the job.
```

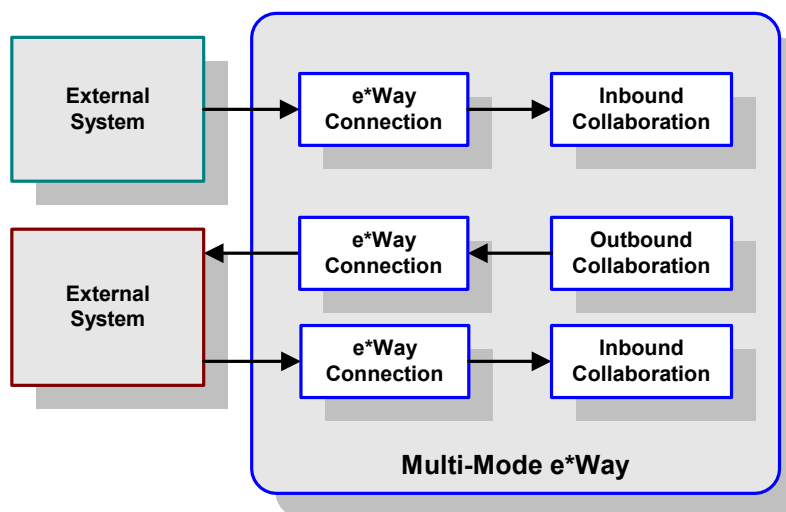


5.5 Basic e*Gate/e*Way Operation

5.5.1 Multi-Mode e*Way

The Siebel EIM e*Way is based on the SeeBeyond Multi-Mode e*Way, which is a multi-threaded component forming an Intelligent Adapter for e*Gate Integrator to exchange information with multiple external systems. The e*Way connects to one or more external systems by means of *e*Way Connections*, each of which must be configured for the specific external system to which it connects (see Figure 52).

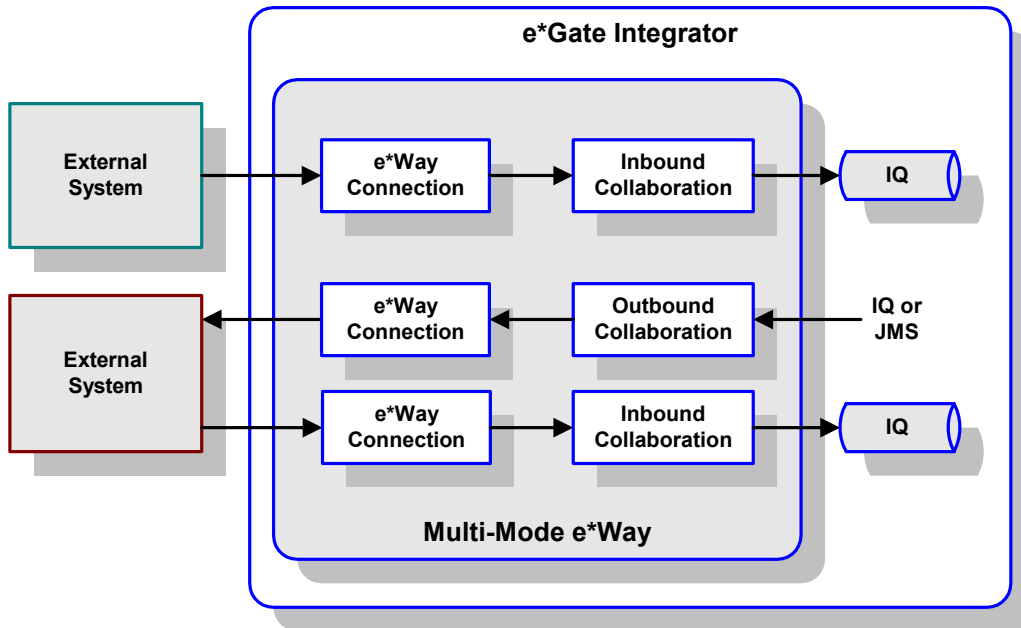
Figure 52 Multi-Mode e*Way



Each e*Way performs one or more *Collaborations* (see [Collaborations](#) on page 100). Bidirectional data flow requires at least two Collaborations, one *Inbound* and one *Outbound*, as shown in Figure 52. Each Collaboration processes a stream of messages, or *Events*, containing data or other information.

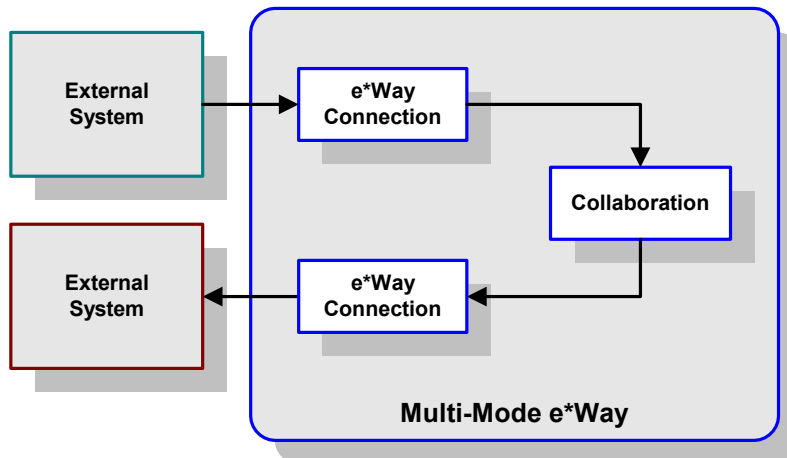
Each Collaboration that publishes its processed Events internally (within e*Gate Integrator) requires one or more *Intelligent Queues* (IQs) to receive the Events (see Figure 53). Any Collaboration that publishes its processed Events only to an external system *does not* require an IQ to receive Events.

Figure 53 e*Way within e*Gate Integrator



Although usually implemented as above, this e*Way also can be implemented as a stand-alone bridge between two or more external systems (see Figure 54).

Figure 54 Stand-alone e*Way



5.5.2 Event Type Definitions

ETDs can be categorized in two types:

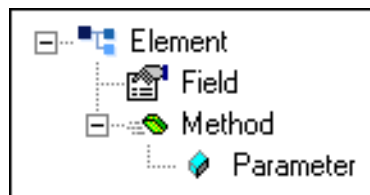
- *Data* ETDs, which represent the structure of Events (messages)
- *API-based* ETDs, or *Complex* ETDs, primarily contain APIs for communicating with external systems

Transforming data from one format to another is a major part of the processing performed by the e*Way, and data ETDs represent the data structure required by specific external systems. Generally, these ETDs are referred to as being *marshalable* because they can be *marshaled* into a flat, non hierarchical structure. See the *e*Gate Integrator User's Guide* for an extensive explanation of ETDs.

ETD Components

There are four possible components (nodes) in the Java Event Type Definition as shown in Figure 55.

Figure 55 The Java-enabled ETD

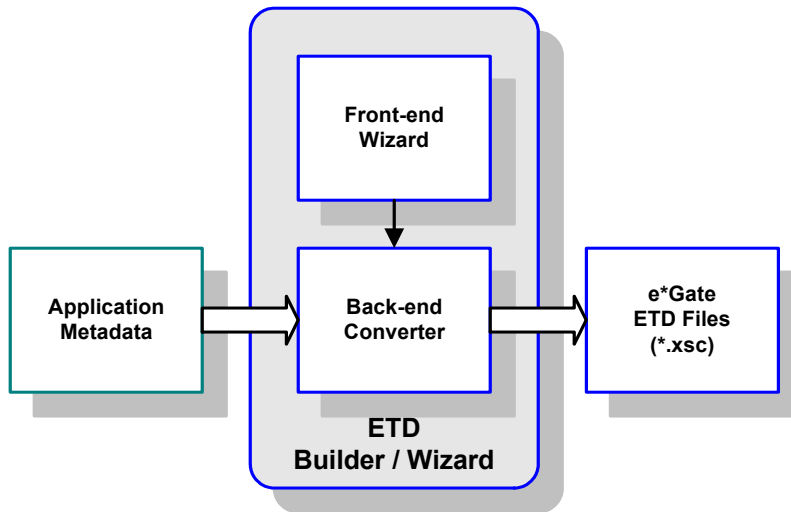


- *Elements* are the basic containers that holds the other parts of the ETD, and can contain fields and methods
- *Fields* are used to represent data, which can be in any of the following formats:
 - string
 - boolean
 - integer
 - double
 - float
- *Method* nodes represent actual Java methods
- *Parameter* nodes represent the Java method's parameters

ETD Builders

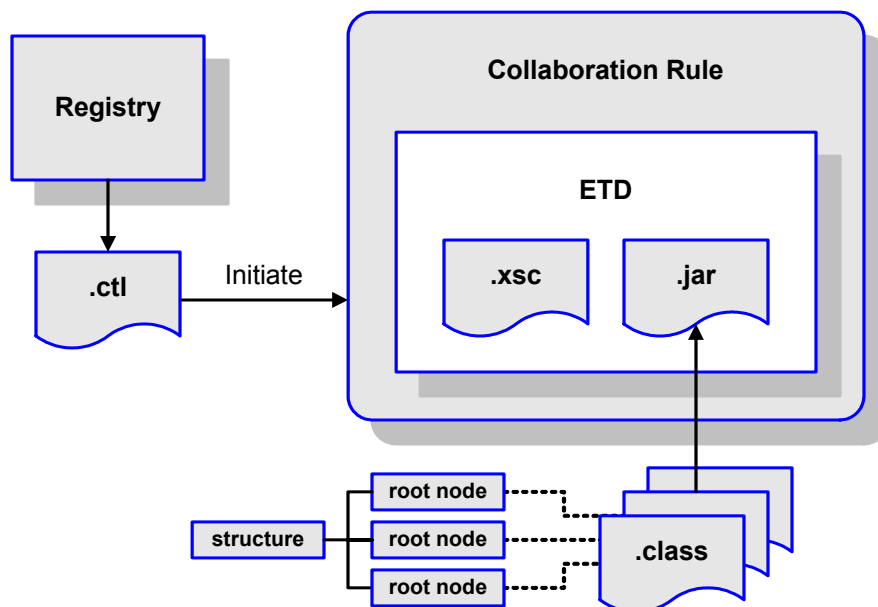
Building an ETD obviously requires knowledge of the internal data structure of the specific application. This information is obtained by extracting metadata from the Siebel database, which is automated by using the *DB Wizard* (see Figure 56).

Figure 56 Typical ETD Wizard



Once compiled, an ETD has two components, an `.xsc` file and a `.jar` file, both having the same file name. The `.jar` file contains `.class` files whose names correspond to the root node names in the ETD. Ultimately, the ETD is used within a Collaboration Rule to define the structure of the corresponding Event. At run time, the Collaboration Rule is initiated according to information contained in a `.ctl` file contained in the e*Gate Registry (see Figure 57).

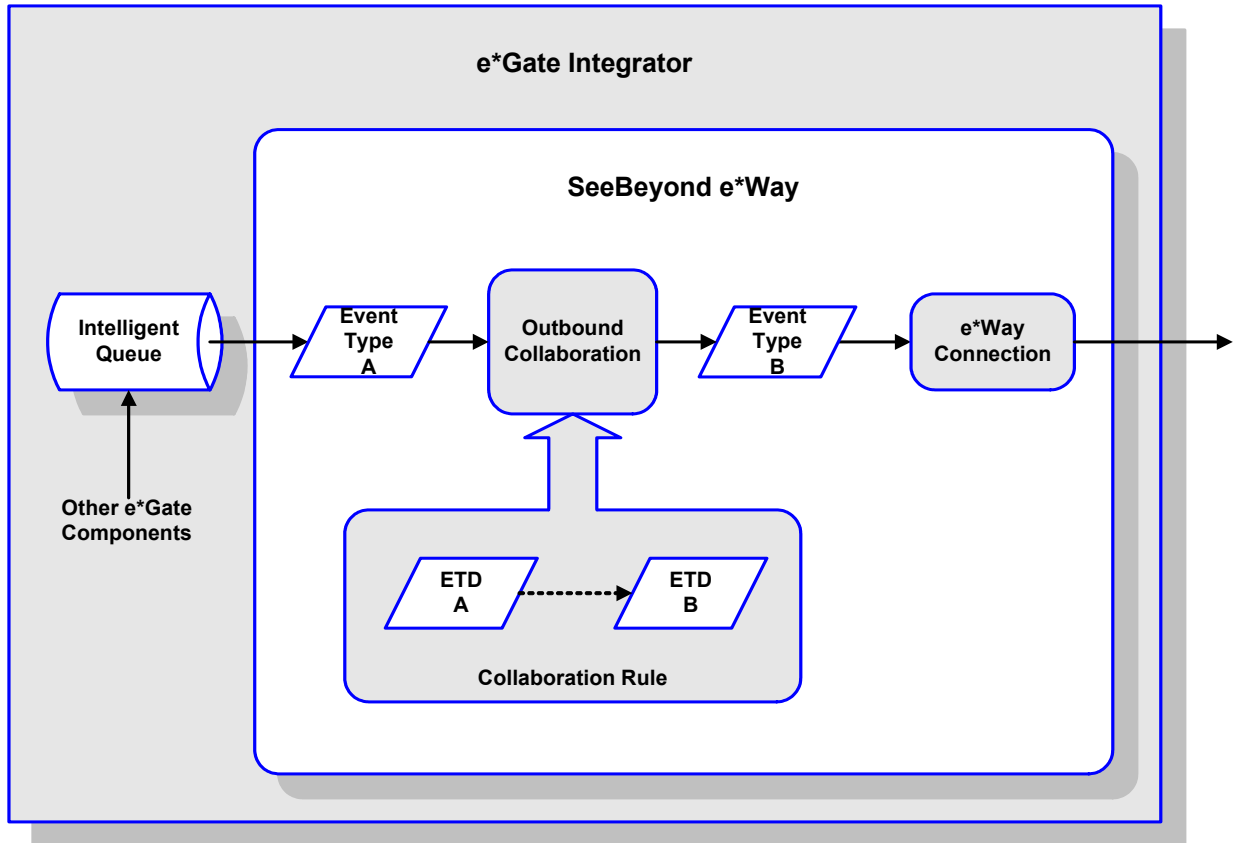
Figure 57 Event Type Definitions



5.5.3 Collaborations

Collaborations execute the business logic that enable the e*Way to perform its intended task. Each Collaboration executes a specified *Collaboration Rule*, which contains the actual instructions to execute the business logic and specifies the applicable *Event Type Definitions* (ETDs). Events Types represent *instances* of their corresponding ETDs. A look inside a typical outbound Collaboration is shown in Figure 58.

Figure 58 Outbound Collaboration

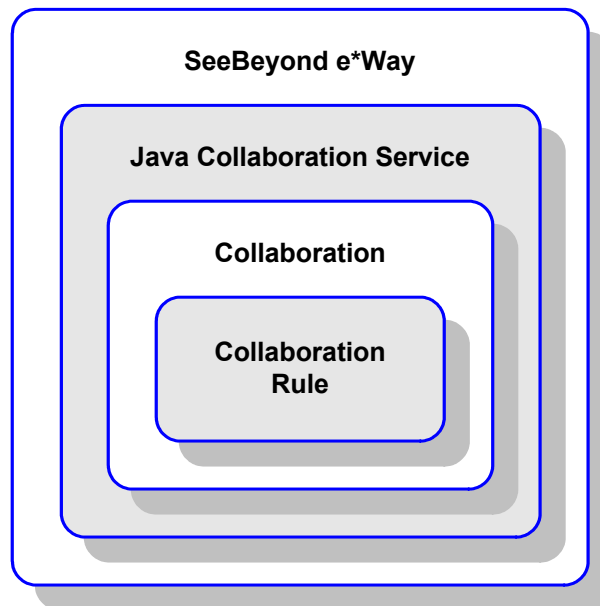


5.5.4 Java Collaboration Service

The Java Collaboration Service (JCS) provides an environment that allows you to use a Java class to implement the business logic that transforms Events as they move through e*Gate. When data passes through e*Gate using a Java Collaboration, a Java Virtual Machine (JVM) is instantiated and uses the associated Java Collaboration Rules class to accomplish the data transformation.

The relationships between the various Java e*Way components can be depicted as a nested structure, as shown in Figure 59.

Figure 59 Java Component Relationships



The Java Collaboration Service makes it possible to develop Collaboration Rules that execute e*Gate business logic using Java code. Using the Java Collaboration Editor, you create Java classes that utilize the `executeBusinessRules`, `userInitialize`, and `userTerminate` methods.

To use the Java Collaboration Service, you create a Collaboration Rule and select Java as the service. Using Event Type instances of previously defined Event Type Definitions (ETDs), you then use the Java Collaboration Rules Editor to add the rules and logic between the Event Type instances. Compiling the Collaboration Rule creates a Java Collaboration Rules class and all required supporting files. This Java class implements the data transformation logic.

For more information on the Java Collaboration Service, see the *e*Gate Integrator Collaboration Services Reference Guide*.

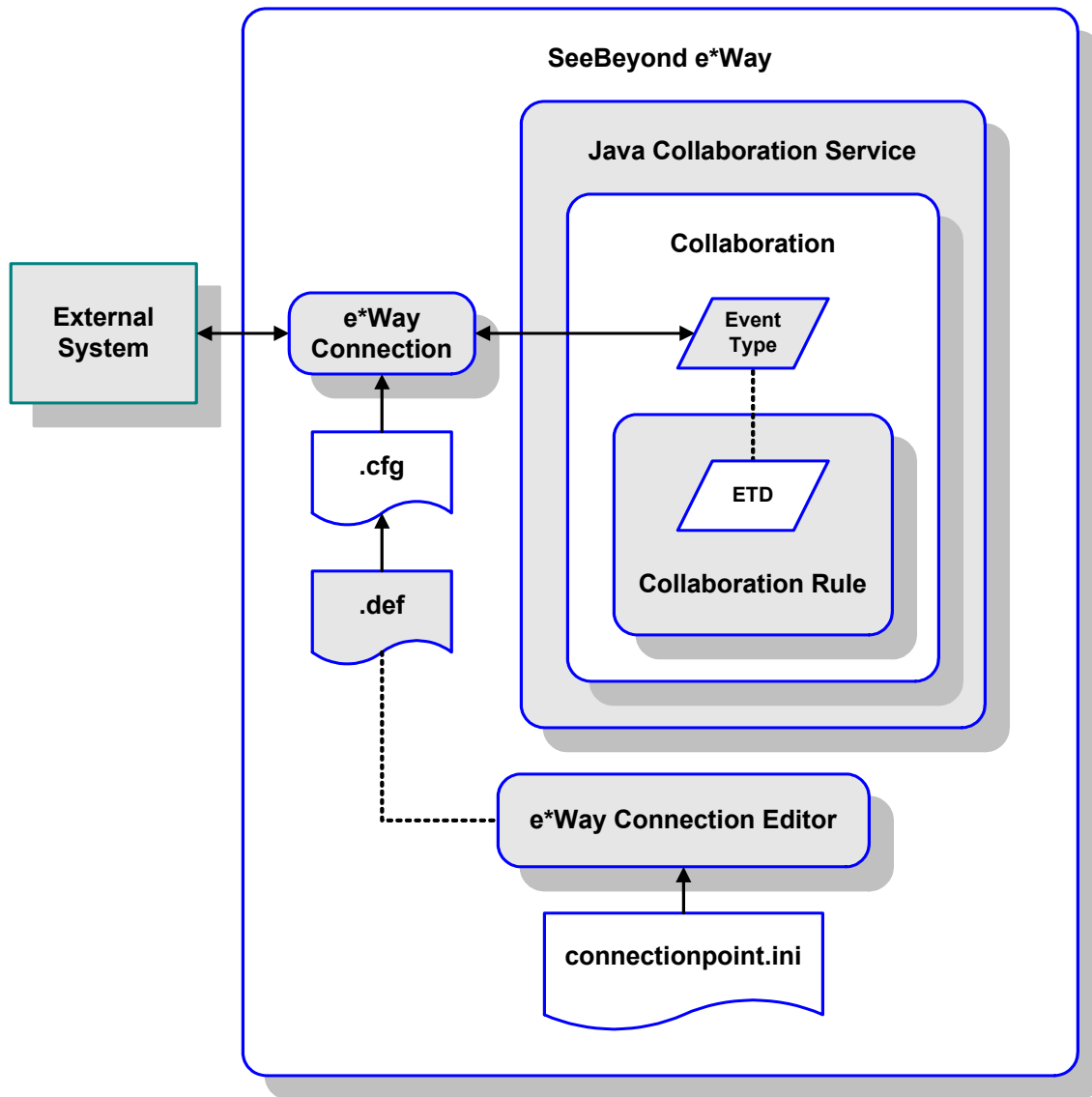
5.5.5 e*Way Connections

The e*Way Connections provide portals to external systems, allowing a single e*Way to adopt several configuration profiles simultaneously. Individual e*Way Connections can be configured using the e*Way Connection Editor to establish a particular kind of interaction with the external system.

Establishing Connections

An e*Way Connection to an external application is set up as depicted in Figure 60. The .def file supplied with the e*Way is configured for the specific application using the e*Way Connection Editor, and instantiated as a .cfg file for each e*Way Connection.

Figure 60 e*Way Connection Establishment



The e*Way Connection Editor enables you to modify all parameters of an e*Way that control the way the e*Way communicates with an external application. Because each e*Way functions in a specific way to provide an interface to a specific external application or communications protocol, each e*Way Connection has a unique set of configuration parameters.

For more information on the Java ETD Editor and the Java Collaboration Editor, see the *e*Gate Integrator User's Guide*.

Configuration Parameters

This chapter describes the configuration parameters for the Java Siebel EIM e*Way.

6.1 Overview

6.1.1 Multi-Mode e*Way

The e*Way's inherent configuration parameters are set using the e*Way Configuration Editor; see [Configuring the e*Way](#) on page 57 for procedural information. The default configuration is provided in `stceway.def`. The Siebel EIM e*Way's Multi-Mode configuration parameters are organized into the following sections:

[JVM Settings](#) on page 106

[General Settings](#) on page 111

For additional information on the Multi-Mode e*Way, see the *Standard e*Way Intelligent Adapter User's Guide*

6.1.2 e*Way Connections

An e*Way Connection's configuration parameters also are set using the e*Way Configuration Editor; see [Creating e*Way Connections](#) on page 62 for procedural information. There are e*Way Connections for both Siebel EIM and the appropriate RDBMS.

Siebel EIM

The default configuration is provided in `SiebelEIM.def`. The Siebel EIM e*Way Connection's configuration parameters are organized into the following sections:

[Connector](#) on page 112

[Siebel Server](#) on page 113

[Siebel Control](#) on page 116

[Siebel System Import](#) on page 117

[Siebel System Export](#) on page 118

JDBC Oracle

The default configuration is provided in **oracle.def**. The Oracle e*Way Connection's configuration parameters are organized into the following sections:

DataSource Settings on page 119

Connector Settings on page 122

For additional information on the Oracle e*Way, see the *e*Way Intelligent Adapter for Oracle User's Guide*

DB2 Universal Database

The default configuration is provided in **db2.def**. The DB2 e*Way Connection's configuration parameters are organized into the following sections:

DataSource Settings on page 124

Connector Settings on page 128

For additional information on the DB2 e*Way, see the *e*Way Intelligent Adapter for DB2 Universal Database User's Guide*

Microsoft SQL Server

The default configuration is provided in **sqlserver.def**. The SQL Server e*Way Connection's configuration parameters are organized into the following sections:

DataSource Settings on page 130

Connector Settings on page 133

For additional information on the SQL Server e*Way, see the *e*Way Intelligent Adapter for SQL Server User's Guide*

6.2 Multi-Mode e*Way

6.2.1 JVM Settings

The JVM Settings control basic Java Virtual Machine settings.

JNI DLL Absolute Pathname

Description

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

Required Values

A valid pathname.

Note: This parameter is **required**, and must **not** be left blank.

Additional Information

The JNI DLL name varies for different operating systems:

Operating System	Java 2 JNI DLL Name
Windows 2000	jvm.dll
Solaris 2.6, 2.7, 2.8	libjvm.so
Linux 6	libjvm.so
HP-UX	libjvm.sl
AIX 4.3	libjvm.a

The value assigned can contain a reference to an environment variable, by enclosing the variable name within a pair of % symbols. For example:

```
%MY_JNIDLL%
```

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

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

CLASSPATH Prepend

Description

Specifies the paths to be prefixed to the CLASSPATH environment variable for the Java VM.

Required Values

An absolute path or an environmental variable.

Note: This parameter is optional and may be left blank.

Additional Information

If left blank, no paths will be prefixed to the CLASSPATH environment variable.

Existing environment variables may be referenced in this parameter by enclosing the variable name in a pair of % signs. For example:

```
%MY_PRECLASSPATH%
```

CLASSPATH Override

Description

Specifies the complete CLASSPATH variable to be used by the Java VM. This parameter is optional. If left blank, 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 both e*Gate and the Java VM must be included. It is advised that the **CLASSPATH Prepend** parameter should be used.

Required Values

An absolute path or an environment variable.

Note: This parameter is optional and may be left blank.

Additional Information

Existing environment variables may be referenced in this parameter by enclosing the variable name in a pair of % signs. For example:

```
%MY_CLASSPATH%
```

CLASSPATH Append From Environment Variable

Description

Specifies whether to attach the environment variable to the end of CLASSPATH.

Required Values

YES or NO. The default value is NO.

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 Java VM will be used.

Required Values

An integer between 0 and 2147483647.

Note: This parameter is optional and may be left blank.

Maximum Heap Size

Description

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

Required Values

An integer between 0 and 2147483647.

Note: This parameter is optional and may be left blank.

Maximum Stack Size for Native Threads

Description

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

Required Values

An integer between 0 and 2147483647.

Note: This parameter is optional and may be left blank.

Maximum Stack Size for JVM Threads

Description

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

Required Values

An integer between 0 and 2147483647.

Note: This parameter is optional and may be left blank.

Class Garbage Collection

Description

Specifies whether the Class Garbage Collection will be done automatically by the Java VM. The selection affects performance issues.

Required Values

YES or NO.

Garbage Collection Activity Reporting

Description

Specifies whether garbage collection activity will be reported for debugging purposes.

Required Values

YES or NO.

Asynchronous Garbage Collection

Description

Specifies whether asynchronous garbage collection activity will be reported for debugging purposes.

Required Values

YES or NO.

Report JVM Info and all Class Loads

Description

Specifies whether the JVM information and all class loads will be reported for debugging purposes.

Required Values

YES or NO.

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 whether to allow remote debugging of the JVM.

Required Values

YES or NO.

Suspend option for debugging

Description

Specifies whether to suspend option for debugging on JVM startup.

Required Values

YES or NO.

6.2.2 General Settings

Rollback Wait Interval

Description

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

Required Values

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

6.3 Siebel EIM e*Way Connections

6.3.1 Connector

The Connector settings define the high level characteristics of the e*Way Connection.

type

Description

Specifies the type of e*Way Connection. The default value always should be used.

Required Values

The default is **siebeleim**.

class

Description

Specifies the class name of the **siebeleim** connector object.

Required Values

The default is **com.stc.eways.siebeleim.SiebelEIMConnector**.

6.3.2 Siebel Server

The parameters in this section initialize the Siebel Server variables.

Siebel Version

Description

Specifies the Siebel version used to connect with e*Gate

Required Values

One of the following options:

- Siebel7
- Siebel6

The default is Siebel7.

Database Client Connectivity

Description

Specifies the connection type version used to connect to the database.

Required Values

One of the following options:

- DB2
- ORACLE
- SQL Server

The default is Oracle.

Database Host Name

Description

Specifies the host machine where the Siebel database resides. If using SQL Server, this is the data source name corresponding to the Siebel database.

Required Values

A string—there is no default value.

Database Table Owner

Description

Specifies the Siebel database logon name that owns the tables being operated on.

Required Values

A string—there is no default value.

Database User Name

Description

Specifies the Siebel database logon ID.

Required Values

A string—there is no default value.

Database Password

Description

Specifies the Siebel database password.

Required Values

An encrypted string—there is no default value.

Gateway Server

Description

Specifies the address or name of the Gateway Server platform.

Required Values

A string—there is no default value.

Enterprise Server Name

Description

Specifies the Enterprise Server name.

Required Values

A string—there is no default value.

Administrator User Name

Description

Specifies the Administrator user name.

Required Values

A string—there is no default value.

Administrator Password

Description

Specifies the Administrator password.

Required Values

An encrypted string—there is no default value.

Siebel Server Name

Description

Specifies the name of the Siebel Server you are using.

Required Values

A string—there is no default value.

Path to Siebel Server

Description

Specifies the *fully-qualified* path to the Siebel Server root directory.

Required Values

A pathname—the default value is \\SiebelBox\Siebel.

6.3.3 Siebel Control

The functions and parameters in this section are used in processing Siebel data.

Batch Number

Description

Specifies the batch number to be used in the IFB file for processing the data.

Required Values

A number—the default is **1200**.

IFB Filename

Description

Specifies the name of the IFB file.

Required Values

A filename—the default is `\\SiebelBox\Siebel\admin\myeim.ifb`. Replace the placeholder values with the actual names.

6.3.4 Siebel System Import

The parameters in this section initialize the Siebel Server variables for Import only.

Siebel State File

Description

Determines the state of the inbound message at the end of the last run.

Required Values

A pathname—the default is `data\SiebelEIM\import_state.txt`.

Additional Information

Upon startup the Siebel e*Way checks this file. If no file is found, the state is initialized to `MSG_ACKED`. The Siebel State File accepts a relative path (e.g., `data\siebel.doc`) relative to `eGate\client`.

There are three possible states:

State 0 - `MSG_ACKED`

This is the initial state used for new incoming messages. It also represents the final state in which the data has been inserted into the interface table, processed by the Enterprise Interface Manager and the next message is processed into the queue.

State 1 - `MSG_INSERTED`

In this state the message has been inserted into the interface table but no acknowledgment has been sent to e*Gate

State 2 - `MSG_PROCESSED`

In this state the message has been inserted into the interface table and processed by the Enterprise Interface Manager, but no acknowledgment has been sent to e*Gate.

Note: *The integrity of this file is crucial to maintaining the correct state of the import; therefore, editing of this file is not recommended.*

6.3.5 Siebel System Export

The parameters in this section initialize the Siebel Server parameters for Export only.

Last Time of Export File

Description

Specifies the file that holds the time of when the last export was done.

Required Values

A pathname; the default is `data\SiebelEIM\last_exported_time.txt`.

Additional Information

The integrity of this file is crucial to maintaining the correct time of when the last export was done in the Siebel e*Way; therefore, editing of this file is not recommended.

Siebel State File

Description

Determines the state of the outbound Event at the end of the last run.

Required Values

A pathname; the default is `data\SiebelEIM\export_state.txt`.

Note: The State is not currently implemented for Export.

6.4 JDBC Oracle e*Way Connections

6.4.1 DataSource Settings

The DataSource settings define the parameters used to interact with the external database.

class

Description

Specifies the Java class in the JDBC driver that is used to implement the **ConnectionPoolDataSource** interface.

Required Values

A valid class name; the default is **oracle.jdbc.pool.OracleConnectionPoolDataSource**.

Additional Information

Use the **oracle.jdbc.xa.client.OracleXADataSource** class for XA-compliant implementations.

To use XA in Oracle 8i and Oracle9i, your Database Administrator needs to GRANT SELECT ON DBA_PENDING_TRANSACTIONS TO PUBLIC.

Note: XA functionality is not supported on Oracle 8.0.5 databases. For more information on implementing the Oracle e*Way in an XA compliant environment, see the *e*Gate Integrator User's Guide*.

DriverType

Description

Specifies the JDBC driver type for Oracle (all other JDBC drivers are ignored).

Oracle implicitly issue a commit statement even if auto commit is set to false and no explicit commit or rollback is executed. Please see Chapter 3 of the *JDBC Developer's Guide and Reference* for Oracle 8.1.7.

Any DDL operation, such as CREATE or ALTER, always includes an implicit COMMIT. If auto-commit mode is disabled, this implicit COMMIT will not only commit the DDL statement, but also any pending DML operations that had not yet been explicitly committed or rolled back.

Required Values

Either **thin** or **oci8**; the default is **thin**.

ServerName

Description

Specifies the host name of the external database server.

Required Values

Any valid string.

PortNumber

Description

Specifies the I/O port number on which the server is listening for connection requests.

Required Values

A valid port number; the default is 1521.

DatabaseName

Description

Specifies the name of the database instance.

Required Values

Any valid string.

user name

Description

Specifies the user name the e*Way will use to connect to the database.

Required Values

Any valid string.

password

Description

Specifies the password used to access the database.

Required Values

Any valid string.

timeout

Description

Specifies the login timeout in seconds.

Required Values

Any valid string; the default is **300** seconds.

6.4.2 Connector Settings

The Connector settings define the high level characteristics of the e*Way Connection. See [Managing e*Way Connections](#) on page 66 for additional information.

type

Description

Specifies the type of e*Way Connection.

Required Values

The default is **DB**.

class

Description

Specifies the class name of the JDBC connector object.

Required Values

The default is **com.stc.eways.jdbcx.DbConnector**.

transaction mode

This parameter specifies how a transaction should be handled.

- **Automatic** — e*Gate will take care of transaction control and users should not issue a commit or rollback. If you are using XA, you must set your **connection establishment mode** and your **transaction mode** both to **Automatic**.
- **Manual** — You will manually take care of transaction control by issuing a commit or rollback.

Required Values

The required values are **Automatic** or **Manual**. The default is set to **Automatic**.

connection establishment mode

This parameter specifies how a connection with the database server is established and closed.

- **Automatic** indicates that the connection is automatically established when the collaboration is started and keeps the connection alive as needed. If you are using XA, you must set your **connection establishment mode** and your **transaction mode** both to **Automatic**.
- **OnDemand** indicates that the connection will be established on demand as business rules requiring a connection to the external system are performed. The connection will be closed after the methods are completed.

- **Manual** indicates that the user will explicitly call the connection connect and disconnect methods in their collaboration as business rules.

Required Values

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

Note: If you are using Manual connection establishment mode, you must also use Manual transaction mode.

connection inactivity timeout

This value is used to specify the timeout for the Automatic connection establishment mode.

- If this is set to 0, the connection will not be brought down due to inactivity. The connection is always kept alive; if it goes down, re-establishing the connection will automatically be attempted.
- If a non-zero value is specified, the connection manager will try to monitor for inactivity so that the connection is brought down if the timeout specified is reached.

Required Values

Any valid string.

connection verification interval

This value is used to specify the minimum period of time between checks for connection status to the database server. If the connection to the server is detected to be down during verification, your collaboration's **onDown** method is called. If the connection comes up from a previous connection error, your collaboration's **onUp** method is called.

Required Values

Any valid string.

6.5 DB2 UDB e*Way Connections

6.5.1 DataSource Settings

The DataSource settings define the parameters used to interact with the external database.

class

Description

Specifies the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.

Required Values

A valid class name; the default is **com.SeeBeyond.jdbcx.db2.DB2DataSource**.

Connection Method

Specifies which method is used to connect to the database server:

- **Pooled Data Source**

A ConnectionPoolDataSource object for creating PooledConnection objects. A PooledConnection object represents a physical connection and is cached in memory for reuse, which saves the overhead of establishing a new connection. This is implemented by the driver.

- **XA Data Source**

An XADataSource object for creating XAConnection objects, which are connections that can be used for distributed transaction.

You should make sure that the class specified in the **class** parameter supports the connection method that is used here.

Required Values

The default is **Pooled Data Source**.

CollectionID

Description

The collection or group of packages to which a package is bound.

Required Values

Any valid string.

DatabaseName

Description

Specifies the name or alias of the database instance. The alias must match the database alias in the DB2 client's database directory. If you are using IBM's DB2 JDBC driver, you will need to enter the database alias name that is configured on the DB2 client.

Required Values

Any valid string.

LocationName

Description

The name of the DB2 location that you want to access.

Required Values

Any valid string.

PackageName

Description

Specifies the name, 7 character limit, of the package that the drive uses to process static and dynamic SQL.

Required Values

Any valid string.

user name

Description

Specifies the case-insensitive user name the e*Way uses to connect to the database.

Required Values

Any valid string.

password

Description

Specifies the password used to access the database.

Required Values

Any valid string.

PortNumber

Description

The TCP port number.

Note: *PortNumber* is used for *DataSource* connections only.

Required Values

Any valid string.

ServerName

Description

The IP address used for *DataSource* connections only. If you are using IBM's DB2 JDBC driver enter the name of your local machine.

- If you are using the IBM DB2 Connect JDBC driver and no *Hostname* is specified, then the IBM DB2 Connect JDBC Type 2 driver will be invoked.
- If you are using the IBM DB2 Connect JDBC driver and a *Hostname* is specified, then the IBM DB2 Connect JDBC Type 3 driver will be invoked.

Note: *XA* is only supported with the IBM DB2 Connect JDBC Type 2 driver.

Required Values

Any valid string.

timeout

Description

Specifies the login timeout in seconds.

Required Values

Any valid string; the default is 300 seconds.

data source attribute value pair separator

Description

This entry specifies the character separator used to separate the attribute-value pair used in [data source attributes](#) on page 127. For example, the attribute-value pair **ServerName!myHost** has **<!>** as a separator.

One should select a separator that will *not* be part of the attribute-name or the attribute-value.

Required Values

The default value is <!.>.

data source attributes

A list of separated attribute-value pairs. This information is used to identify the database and set the connection properties. The attribute name should be exactly the same as the one that is specified in the driver documentation and the value should be a valid one. The whole list will be used to specify the connection properties. To disable an attribute, simply un check it.

For example: **PortNumber! 8888**

The separator used in this parameter should match the one specified in [data source attribute value pair separator](#) on page 126. By default, the separator used is <!.>.

Required Values

Any separated attribute-value pair.

6.5.2 Connector Settings

The Connector settings define the high-level characteristics of the e*Way Connection.

type

Description

Specifies the type of e*Way Connection.

Required Values

The default is **DB**.

class

Description

Specifies the JAVA class name of the JDBC connector object.

Required Values

The default is **com.stc.eways.jdbcx.DbConnector**.

transaction mode

This parameter specifies how a transaction should be handled.

- **Automatic** — e*Gate will take care of transaction control and users should not issue a commit or rollback. If you are using XA, you must set your **connection establishment mode** and your **transaction mode** both to **Automatic**.
- **Manual** — You will manually take care of transaction control by issuing a commit or rollback.

Required Values

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

connection establishment mode

This parameter specifies how a connection with the database server is established and closed.

- **Automatic** indicates that the connection is automatically established when the collaboration is started and keeps the connection alive as needed. If you are using XA, you must set your **connection establishment mode** and your **transaction mode** both to **Automatic**.
- **OnDemand** indicates that the connection will be established on demand as business rules requiring a connection to the external system are performed. The connection will be closed after the methods are completed.

- **Manual** indicates that the user will explicitly call the connection connect and disconnect methods in their collaboration as business rules.

Required Values

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

Note: If you are using Manual connection establishment mode, you must also use Manual transaction mode.

connection inactivity timeout

This value is used to specify the timeout for the Automatic connection establishment mode.

- If this is set to 0, the connection will not be brought down due to inactivity. The connection is always kept alive; if it goes down, re-establishing the connection will automatically be attempted.
- If a non-zero value is specified, the connection manager will try to monitor for inactivity so that the connection is brought down if the timeout specified is reached.

Required Values

Any valid string.

connection verification interval

This value is used to specify the minimum period of time between checks for connection status to the database server. If the connection to the server is detected to be down during verification, your collaboration's **onDown** method is called. If the connection comes up from a previous connection error, your collaboration's **onUp** method is called.

Required Values

Any valid string.

6.6 SQL Server e*Way Connections

6.6.1 DataSource Settings

The DataSource settings define the parameters used to interact with the external database.

class

Description

Specifies the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.

Required Values

A valid class name; the default is `com.SeeBeyond.jdbcx.sqlserver.SQLServerDataSource`.

Connection Method

Specifies which method is used to connect to the database server:

- **Pooled Data Source**

A ConnectionPoolDataSource object for creating PooledConnection objects. A PooledConnection object represents a physical connection and is cached in memory for reuse, which saves the overhead of establishing a new connection. This is implemented by the driver.

- **XA Data Source**

An XADataSource object for creating XAConnection objects, which are connections that can be used for distributed transaction.

You should make sure that the class specified in the **class** parameter supports the connection method that is used here.

Required Values

The default is **Pooled Data Source**.

ServerName

Description

Specifies the host name of the external database server.

Required Values

Any valid string.

PortNumber

Description

Specifies the port number on which the server is listening for connection requests. You must specify a valid port number. This e*Way does not support dynamic port assignments.

Required Values

A valid port number; the default value is **1433**.

DatabaseName

Description

Specifies the name of the database instance.

Required Values

Any valid string.

user name

Description

Specifies the user name the e*Way will use to connect to the database.

Required Values

Any valid string.

password

Description

Specifies the password used to access the database.

Required Values

Any valid string.

SelectMethod

Description

Specifies whether database cursors are used for Select statements. Using the Direct mode is the most efficient for executing Select statements however, you may be limited to a single active statement while executing inside a transaction. Setting SelectMethod to cursor allows multiple active statements within a transaction.

Required Values

Cursor or **Direct**

timeout

Description

This is the login timeout in seconds.

Required Values

Any valid string. The default is **300** seconds.

6.6.2 Connector Settings

The Connector settings define the high-level characteristics of the e*Way Connection.

type

Description

Specifies the type of e*Way Connection. The current available type for JDBC connections is **DB**.

Required Values

The default is **DB**.

class

Description

Specifies the class name of the JDBC connector object.

Required Values

The default is **com.stc.eways.jdbcx.DbConnector**.

transaction mode

This parameter specifies how a transaction should be handled.

- **Automatic** — e*Gate will take care of transaction control and users should not issue a commit or rollback. If you are using XA, you must set your **connection establishment mode** and your **transaction mode** both to **Automatic**.
- **Manual** — You will manually take care of transaction control by issuing a commit or rollback.

Required Values

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

connection establishment mode

This parameter specifies how a connection with the database server is established and closed.

- **Automatic** indicates that the connection is automatically established when the collaboration is started and keeps the connection alive as needed. If you are using XA, you must set your **connection establishment mode** and your **transaction mode** both to **Automatic**.
- **OnDemand** indicates that the connection will be established on demand as business rules requiring a connection to the external system are performed. The connection will be closed after the methods are completed.

- **Manual** indicates that the user will explicitly call the connection connect and disconnect methods in their collaboration as business rules.

Required Values

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

Note: If you are using Manual connection establishment mode, you must also use Manual transaction mode.

connection inactivity timeout

This value is used to specify the timeout for the Automatic connection establishment mode.

- If this is set to 0, the connection will not be brought down due to inactivity. The connection is always kept alive; if it goes down, re-establishing the connection will automatically be attempted.
- If a non-zero value is specified, the connection manager will try to monitor for inactivity so that the connection is brought down if the timeout specified is reached.

Required Values

Any valid string.

connection verification interval

This value is used to specify the minimum period of time between checks for connection status to the database server.

- If the connection to the server is detected to be down during verification, your collaboration's **onDown** method is called.
- If the connection comes up from a previous connection error, your collaboration's **onUp** method is called.

Required Values

Any valid string.

Java Classes and Methods

The Java e*Way Intelligent Adapter for Siebel EIM contains Java methods that are used to extend the functionality of the basic e*Way core.

7.1 Overview

This chapter contains descriptions of methods that are exposed in the user interface. Additional methods contained in the e*Way should only be accessed or modified by qualified SeeBeyond personnel. Unless otherwise noted, all classes and methods described in this chapter are **public**.

[Object Classes](#) on page 136

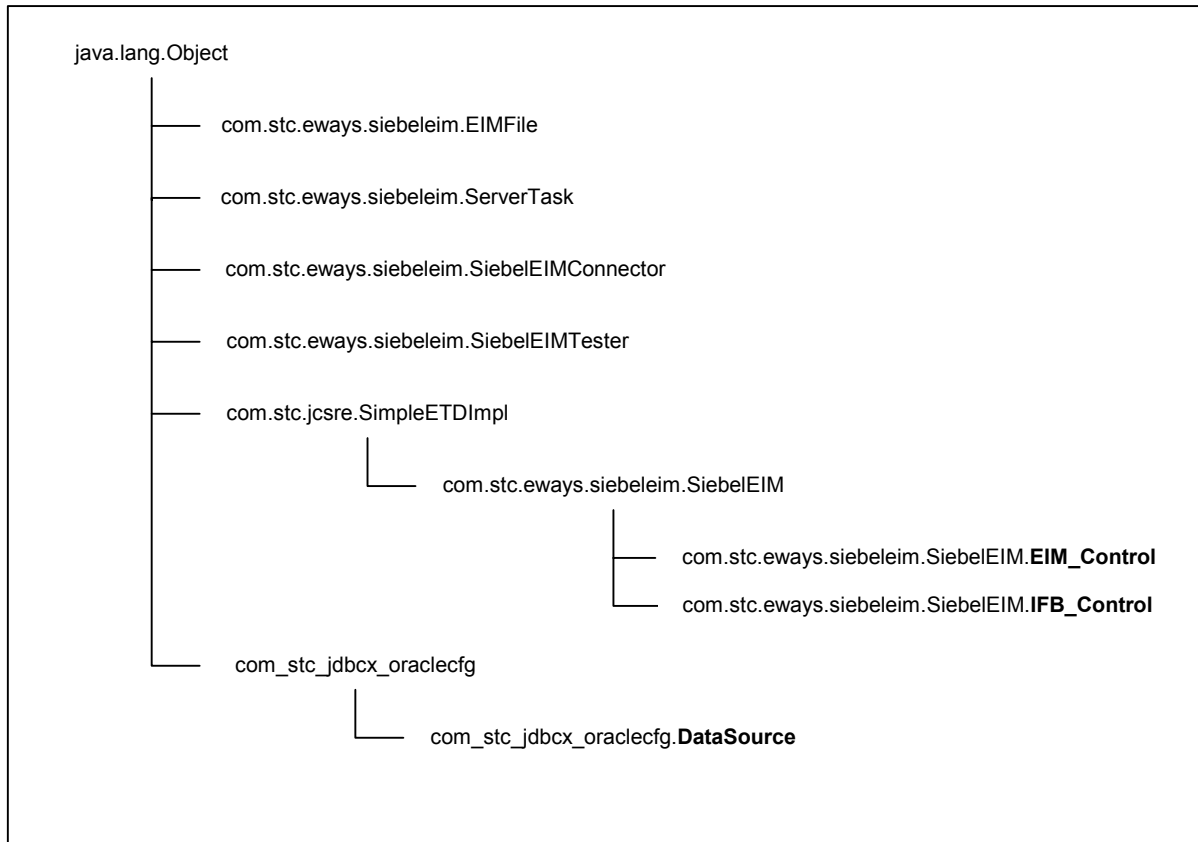
[EIM_Control Class](#) on page 137

[IFB_Control Class](#) on page 141

7.2 Object Classes

The Java Siebel EIM e*Way object methods are contained in the following classes:

Figure 61 Class Hierarchy



Methods exposed in the ETD Editor are contained in the **EIM_Control** and **IFB_Control** classes.

7.3 EIM_Control Class

Description

Extends `java.lang.Object`

Signature

```
com_stc_eways_siebeleim.SiebelEIM.EIM_Control
```

Methods

[getState](#) on page 137

[getStatus](#) on page 138

[startTask](#) on page 138

[startTaskAndWaitUntilDone](#) on page 139

[waitUntilDone](#) on page 139

7.3.1 Methods

getState

Description

This method gets and returns the Siebel EIM run state associated with the task number.

Signature

```
checkState(taskNum)
```

Parameters

Name	Type	Description
taskNum	java.lang.String	The task number of the Siebel EIM job.

Return Type

java.lang.String

Overrides

None.

Throws

- `com.stc.common.collabService.CollabConnException`
- `com.stc.common.collabService.CollabDataException`
- `com.stc.common.collabService.CollabResendException`

getStatus

Description

This method gets and returns the Siebel EIM status associated with the task number.

Signature

```
checkStatus ( taskNum)
```

Parameters

Name	Type	Description
taskNum	java.lang.String	The task number of the Siebel EIM job.

Return Type

java.lang.String

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

startTask

Description

This method starts the Siebel EIM task and returns the task number of the job.

Signature

```
startTask ()
```

Parameters

None.

Return Type

java.lang.String

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

startTaskAndWaitUntilDone

Description

This method starts the Siebel EIM task and runs the EIM job. It also gets the run state of the task repeatedly every ten seconds until it is finished running. If the task exits with error, it also gets the status of the task.

Signature

```
startTaskAndWaitUntilDone()
```

Parameters

None.

Returns

java.lang.String

- FAILURE
- EIM_SUCCESS
- SHUTDOWN
- EXIT_WITH_ERROR
- UNKNOWN_FAILURE

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

waitUntilDone

Description

This method repeatedly (every ten seconds) gets the run state of the task until it finishes running. If the task exits with error, it also gets the status of the task.

Signature

```
waitUntilDone(taskNum)
```

Parameters

Name	Type	Description
taskNum	java.lang.String	The task number of the Siebel EIM job.

Returns

java.lang.String

- FAILURE
- EIM_SUCCESS
- SHUTDOWN
- EXIT_WITH_ERROR
- UNKNOWN_FAILURE

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

7.4 IFB_Control Class

Description

Extends java.lang.Object

Signature

```
com_stc_eways_siebeleim.SiebelEIM.IFB_Control
```

Methods

[generateIfb](#) on page 141

[getImportState](#) on page 141

[getExportState](#) on page 142

[getLastExportTime](#) on page 143

[setImportState](#) on page 142

[setExportState](#) on page 143

[setLastExportTime](#) on page 144

7.4.1 Methods

generateIfb

Description

This method creates Siebel EIM configuration file at the specified location, returning the absolute path of the EIM configuration file.

Signature

```
generateIfb()
```

Parameters

None.

Return Type

java.lang.String

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

getImportState

Description

This method gets and returns the state from the import state file.

Signature

```
getImportState()
```

Parameters

None.

Return Type

java.lang.String

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

setImportState

Description

This method writes the import state into file.

Signature

```
setImportState(stateStr)
```

Parameters

Name	Type	Description
stateStr	java.lang.String	The import state.

Return Type

void

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

getExportState

Description

This method gets and returns the state from the export state file.

Signature

```
getExportState()
```

Parameters

None.

Return Type

java.lang.String

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

setExportState

Description

This method writes the import state into file.

Signature

```
setExportState(stateStr)
```

Parameters

Name	Type	Description
stateStr	java.lang.String	The export state.

Return Type

void

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

getLastExportTime

Description

Gets the last export time from file.

Signature

```
getLastExportTime()
```

Parameters

None.

Return Type

java.lang.String

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

setLastExportTime

Description

Writes the last export time into file.

Signature

```
setLastExportTime(timeStr)
```

Parameters

Name	Type	Description
timeStr	java.lang.String	The last export time.

Return Type

void

Overrides

None.

Throws

- com.stc.common.collabService.CollabConnException
- com.stc.common.collabService.CollabDataException
- com.stc.common.collabService.CollabResendException

Index

A

Administrator Password parameter 115
 Administrator User Name parameter 114
 Assigning ETDs to Event Types 47
 Asynchronous Garbage Collection parameter 109
 Autorun 20

B

Base Tables 52, 53, 77, 81, 82
 Batch Number parameter 116

C

Changing the User Name 58
 class (Connector) parameter 112
 Class Garbage Collection parameter 109
 class parameter (DB2 Connector Settings) 128
 class parameter (DB2 DataSource Settings) 124
 class parameter (Oracle Connector Settings) 122
 class parameter (Oracle DataSource Settings) 119
 class parameter (SQL Connector Settings) 133
 class parameter (SQL DataSource Settings) 130
 CLASSPATH Append From Environment Variable parameter 107
 CLASSPATH Override parameter 107
 CLASSPATH Prepend parameter 106
 Collaboration 52, 71, 76, 77, 100
 Collaboration Rules script 71, 72
 Collaboration Service 71
 CollectionID parameter (DB2 DataSource Settings) 124
 Components, e*Way 16
 configuration
 Connector 112
 Connector Settings (DB2) 128–129
 Connector Settings (Oracle) 122–123
 Connector Settings (SQL) 133–134
 DataSource Settings (DB2) 124–127
 DataSource Settings (Oracle) 119–121
 DataSource Settings (SQL) 130–132
 General Settings 111
 JVM Settings 106–110
 Siebel Control 116
 Siebel Server 113–115
 Siebel System Export 118
 Siebel System Import 117
 configuration parameters
 Administrator Password 115
 Administrator User Name 114
 Asynchronous Garbage Collection 109
 Batch Number 116
 class (Connector) 112
 class (DB2 Connector Settings) 128
 class (DB2 DataSource Settings) 124
 class (Oracle Connector Settings) 122
 class (Oracle DataSource Settings) 119
 class (SQL Connector Settings) 133
 class (SQL DataSource Settings) 130
 Class Garbage Collection 109
 CLASSPATH Append From Environment Variable 107
 CLASSPATH Override 107
 CLASSPATH Prepend 106
 CollectionID (DB2 DataSource Settings) 124
 connection establishment mode (DB2 Connector Settings) 128
 connection establishment mode (Oracle Connector Settings) 122
 connection establishment mode (SQL Connector Settings) 133
 connection inactivity timeout (DB2 Connector Settings) 129
 connection inactivity timeout (Oracle Connector Settings) 123
 connection inactivity timeout (SQL Connector Settings) 134
 Connection Method (DB2 DataSource Settings) 124
 Connection Method (SQL DataSource Settings) 130
 connection verification interval (DB2 Connector Settings) 129
 connection verification interval (Oracle Connector Settings) 123
 connection verification interval (SQL Connector Settings) 134
 data source attribute value pair separator (DB2 DataSource Settings) 126
 data source attributes (DB2 DataSource Settings) 127
 Database Client Connectivity 113
 Database Host Name 113
 Database Password 114
 Database Table Owner 113
 Database User Name 114
 DatabaseName (DB2 DataSource Settings) 125
 DatabaseName (Oracle DataSource Settings) 120

- DatabaseName (SQL DataSource Settings) 131
 - Disable JIT 109
 - DriverType (Oracle DataSource Settings) 119
 - Enterprise Server Name 114
 - Garbage Collection Activity Reporting 109
 - Gateway Server 114
 - IFB Filename 116
 - Initial Heap Size 108
 - JNI DLL Absolute Pathname 106
 - Last Time of Export File 118
 - LocationName (DB2 DataSource Settings) 125
 - Maximum Heap Size 108
 - Maximum Stack Size for JVM Threads 108
 - Maximum Stack Size for Native Threads 108
 - Name 115
 - PackageName (DB2 DataSource Settings) 125
 - password (DB2 DataSource Settings) 125
 - password (Oracle DataSource Settings) 120
 - password (SQL DataSource Settings) 131
 - Path to Siebel Server 115
 - PortNumber (DB2 DataSource Settings) 126
 - PortNumber (Oracle DataSource Settings) 120
 - PortNumber (SQL DataSource Settings) 131
 - Remote debugging port number 110
 - Report JVM Info and all Class Loads 109
 - Rollback Wait Interval 111
 - SelectMethod (SQL DataSource Settings) 131
 - ServerName (DB2 DataSource Settings) 126
 - ServerName (Oracle DataSource Settings) 120
 - ServerName (SQL DataSource Settings) 130
 - Siebel State File (Export) 118
 - Siebel State File (Import) 117
 - Siebel Version 113
 - Suspend option for debugging 110
 - timeout (DB2 DataSource Settings) 126
 - timeout (Oracle DataSource Settings) 121
 - timeout (SQL DataSource Settings) 132
 - transaction mode (DB2 Connector Settings) 128
 - transaction mode (Oracle Connector Settings) 122
 - transaction mode (SQL Connector Settings) 133
 - type (Connector) 112
 - type (DB2 Connector Settings) 128
 - type (Oracle Connector Settings) 122
 - type (SQL Connector Settings) 133
 - user name (DB2 DataSource Settings) 125
 - user name (Oracle DataSource Settings) 120
 - user name (SQL DataSource Settings) 131
 - configuration procedures 57
 - connection establishment mode parameter (DB2 Connector Settings) 128
 - connection establishment mode parameter (Oracle Connector Settings) 122
 - connection establishment mode parameter (SQL Connector Settings) 133
 - connection inactivity timeout parameter (DB2 Connector Settings) 129
 - connection inactivity timeout parameter (Oracle Connector Settings) 123
 - connection inactivity timeout parameter (SQL Connector Settings) 134
 - Connection Manager 66
 - Connection Method parameter (DB2 DataSource Settings) 124
 - Connection Method parameter (SQL DataSource Settings) 130
 - connection types, JDBC 133
 - connection verification interval parameter (DB2 Connector Settings) 129
 - connection verification interval parameter (Oracle Connector Settings) 123
 - connection verification interval parameter (SQL Connector Settings) 134
 - Connector configuration 112
 - connector objects, JDBC 133
 - Connector Settings (DB2) 128–129
 - Connector Settings (Oracle) 122–123
 - Connector Settings (SQL) 133–134
 - conventions, writing in document 10
 - Creating a Schema 35
- ## D
- data source attribute value pair separator parameter (DB2 DataSource Settings) 126
 - data source attributes parameter (DB2 DataSource Settings) 127
 - Database Client Connectivity parameter 113
 - Database Host Name parameter 113
 - Database Password parameter 114
 - Database Table Owner parameter 113
 - Database User Name parameter 114
 - DatabaseName parameter (DB2 DataSource Settings) 125
 - DatabaseName parameter (Oracle DataSource Settings) 120
 - DatabaseName parameter (SQL DataSource Settings) 131
 - DataSource Settings (DB2) 124–127
 - DataSource Settings (Oracle) 119–121
 - DataSource Settings (SQL) 130–132
 - DB2 e*Way Connection configuration
 - Connector Settings 128–129
 - DataSource Settings 124–127
 - DBWizard 37
 - DBWizard ETD Builder 37
 - Disable JIT parameter 109
 - DriverType parameter (Oracle DataSource Settings)

119

E

e*Way

- Components 16
- configuration 57
- creating 55
- Installation 20
- Properties 56
- Schedules 58
- Startup Options 58
- troubleshooting 71

e*Way Connections 62, 66

EIM_Control Class 137

Enterprise Integration Manager (EIM) 13, 53, 73, 81

Enterprise Server Name parameter 114

Event Type 47

Event Type Definition (ETD) 47, 52, 77

G

Garbage Collection Activity Reporting parameter 109

Gateway Server parameter 114

General Settings configuration 111

generateIfb method 141

getExportState method 142

getImportState method 141

getLastExportTime method 143

getState method 137

getStatus method 138

I

IF_ROW_STAT 53, 82

IFB Filename parameter 116

IFB_Control Class 141

inbound process flow 15, 81

Initial Heap Size parameter 108

Installation

e*Way 20

InstallShield 20

Intelligent Queue (IQ) 48, 53, 71, 81

Interface Tables 52, 53, 77, 81, 82

J

Java methods

generateIfb 141

getExportState 142

getImportState 141

getLastExportTime 143

getState 137

getStatus 138

setExportState 143

setImportState 142

setLastExportTime 144

startTask 138

startTaskAndWaitUntilDone 139

waitUntilDone 139

Java Object Classes

EIM_Control 137

IFB_Control 141

JDBC 37

connection types 133

connector objects 133

JNI DLL Absolute Pathname parameter 106

JVM Settings configuration 106–110

L

Last Time of Export File parameter 118

LocationName parameter (DB2 DataSource Settings) 125

logging options 60

M

Maximum Heap Size parameter 108

Maximum Stack Size for JVM Threads parameter 108

Maximum Stack Size for Native Threads parameter 108

monitoring thresholds 61

Multi-Mode e*Way 16, 96, 104

Multi-Mode e*Way configuration

General Settings 111

JVM Settings 106–110

N

Name parameter 115

O

Oracle e*Way Connection configuration

Connector Settings 122–123

DataSource Settings 119–121

P

PackageName parameter (DB2 DataSource Settings) 125

Participating Host 71

password parameter (DB2 DataSource Settings) 125

password parameter (Oracle DataSource Settings) 120
 password parameter (SQL DataSource Settings) 131
 Path to Siebel Server parameter 115
 PortNumber parameter (DB2 DataSource Settings) 126
 PortNumber parameter (Oracle DataSource Settings) 120
 PortNumber parameter (SQL DataSource Settings) 131
 Properties, e*Way 56
 publish 71

Q

Queues 48

R

Remote debugging port number parameter 110
 Report JVM Info and all Class Loads parameter 109
 Rollback Wait Interval parameter 111

S

Schedules 58
 Schema, creating 35
 SelectMethod parameter (SQL DataSource Settings) 131
 ServerName parameter (DB2 DataSource Settings) 126
 ServerName parameter (Oracle DataSource Settings) 120
 ServerName parameter (SQL DataSource Settings) 130
 setExportState method 143
 setImportState method 142
 setLastExportTime method 144
 Setting Startup Options or Schedules 58
 Siebel Control configuration 116
 Siebel e*Way Connection configuration
 Connector 112
 Siebel Control 116
 Siebel Server 113–115
 Siebel System Export 118
 Siebel System Import 117
 Siebel EIM 13
 Siebel Server configuration 113–115
 Siebel State File (Export) parameter 118
 Siebel State File (Import) parameter 117
 Siebel System Export configuration 118
 Siebel System Import configuration 117
 Siebel Version parameter 113

SQL e*Way Connection configuration
 Connector Settings 133–134
 DataSource Settings 130–132
 startTask method 138
 startTaskAndWaitUntilDone method 139
 Startup Options 58
 subscribe 71
 Suspend option for debugging parameter 110

T

timeout parameter (DB2 DataSource Settings) 126
 timeout parameter (Oracle DataSource Settings) 121
 timeout parameter (SQL DataSource Settings) 132
 transaction mode parameter (DB2 Connector Settings) 128
 transaction mode parameter (Oracle Connector Settings) 122
 transaction mode parameter (SQL Connector Settings) 133
 troubleshooting the e*Way 71
 type (Connector) parameter 112
 type parameter (DB2 Connector Settings) 128
 type parameter (Oracle Connector Settings) 122
 type parameter (SQL Connector Settings) 133

U

User name 58
 user name parameter (DB2 DataSource Settings) 125
 user name parameter (Oracle DataSource Settings) 120
 user name parameter (SQL DataSource Settings) 131

W

waitUntilDone method 139