SeeBeyond™ eBusiness Integration Suite

e*Gate Integrator Upgrade Kit Guide

Release 4.5.3

Version 20021023104019.

Contents

List of Figures 7

List of Tables 8

Chapter 1

Introduction 9

Document Purpose and Scope 9
Intended Audience 9
Organization of Information 10
Writing Conventions 10
Supporting Documents 12
SeeBeyond Web Site 13

Chapter 2

Upgrade Overview 14

Upgrading to e*Gate: Introduction 14
System Requirements 14
Installation 15
Terminology 15
Supported Versions 16
e*Gate Compatibility Components 16
Schema Configuration Utility 17
Creating Proxy e*Ways 18
Creating the Route Table Collaboration Service 19
Creating Event Types and Intelligent Queues 20
Associating a BOB with a Route Table 21
e*Way Configuration File Conversion Utility 23
Utility Operation 23
Supported e*Gate 3.6 e*Ways for Configuration File Conversion 24
FIFO 24
BOBs and the Standard IQ FIFO 26

When FIFO is activated 27
Generating a Multi-Mode e*Way with FIFO 27
Upgrade Methodology 28
Using e*Gate Compatibility Component Tools 28
Substituting Equivalent e*Gate 4.5.3 e*Ways 28
Using Proxy e*Ways 29

Chapter 3
Upgrade Procedure 30
Upgrading to e*Gate 4.5.3: Overview 30
Upgrade Prerequisites 30
Upgrade Procedure 31
Running the stcdgschema Utility 32
Committing Files to the e*Gate Registry 36
Undefining the DGOS Environment Variable 37
Making Configuration Changes to the Proxy e*Way 38
Running dgw2ew.cmd 40
Modifying DataGate Interfaces After Encapsulation 45

Chapter 4
Testing and Troubleshooting 47
Upgrading Configuration Parameters 47
Using the e*Way Editor 47
Changes to all Monk-enabled e*Ways 48
Batch or FTP e*Ways 48
Database Access e*Ways 49
MQ Series e*Ways 49
Changes to e*Way-Specific Monk Functions 50
"%standard->julian" Monk Function is No Longer Valid 50
Loop Syntax Change 50
Database Access e*Way Functions 51
Batch/FTP e*Way Functions 54
MQSeries e*Way Functions 60

Chapter 5
Frequently Asked Questions 64
Introduction: Using These FAQs 64
Upgrade FAQs 64
Handling IQs and IQ Properties 64
Table Mode Scripts 65

Improving Performance and Throughput 66
Files, Scripts, and Monk Functions 66
Proxy e*Ways 67
Upgrading From e*Gate 4.X to e*Gate 4.5.1 or Later 67

Appendix A

Upgrade Operation Checklist 68

Prerequisites 68

Schema Construction 69

Appendix B

Upgrading Monk e*Gate 3.6 to e*Gate 4.5.3 70

Pathing Strictness Increased (~ and %) 70

New Tokens 70

Dot "." as Initial Character 70

64 Bit File Type 70

Non-printable Values 71

Dynamic Parsing of Data 71

Dynamic Extension of Event Definitions 71

Modification of the Monk Parser (Dotted Pair) 71

Delimiters 72

Behavior of Optional Nodes 72
Nodes Generated As Output Nodes 73
Order of Promotional Behavior 73

New Interface Functions 74

Immutability 74

Fixed Node Data Definition 74

HL7 Functions 74

Exception Support 75

The regex Functions 75

New Modifiers 75

Modified Monk Functions 76

Functions No Longer Supported 76

Real Number Precision 78

UTF8 and UTF16 78

Appendix C

Moving Encapsulated Routes to Native 79

Moving to Native Routes: Introduction 79

Moving to Native Routes: Procedure 79

Appendix D

Sample Configuration Files 81

Files: Schema Configuration Utility 81

dart_send.dgt 82

dart_send.dgt.rtb 82

dart_send.dgt.ptb 82

db_send.sc 82

db_send.cfg 86

ProxyDart.ctl 87

Files: e*Way Configuration File Conversion Utility 88

db_send.sc 88

dartRule.txt 102

dart.def 103

db_send.sc 115

db_send.ctl 128

db_send.sc.old 128

Glossary 129

Index 137

List of Figures

Figure 1	Inbound Proxy e*Way Operation	19
Figure 2	Outbound Proxy e*Way Operation	19
Figure 3	Creation of IQs	20
Figure 4	Beginning with Routes in e*Gate 3.6	21
Figure 5	Associates Route with Single BOB in e*Gate 4.5.3	22
Figure 6	e*Way Configuration File Conversion Operation	23
Figure 7	Example of Collaboration Subscribing to Event Type Publisher Pairs	25
Figure 8	Example of Collaboration Subscribing to an IQ	26
Figure 9	Enterprise Manager Main Window	34
Figure 10	Participating Host Properties Dialog Box	35
Figure 11	Enterprise Manager with Components Displayed	35
Figure 12	IQ Manager Properties Dialog Box	36
Figure 13	Enterprise Manager with e*Way Displayed	38
Figure 14	e*Way Properties Dialog Box	39
Figure 15	Enterprise Manager with e*Way Displayed	42
Figure 16	Finding the e*Way Executable File	43
Figure 17	Finding the e*Way Configuration File	44
Figure 18	e*Way Properties Dialog Box Showing the Edit Button	48
Figure 19	New e*Way Configuration Files	81

List of Tables

Table 1	DataGate to e*Gate Terminology	15
Table 2	Option Arguments for stcdgschema Command	17
Table 3	e*Way File Directory Locations	40
Table 4	e*Way Conversion Arguments	41
Table 5	e*Way Executable Files	43
Table 6	Database Access e*Way Functions — e*Gate 3.6.1 and 4.5.3	51
Table 7	Database Access e*Way Functions — e*Gate 3.6.2 and 4.5.3	52
Table 8	Batch e*Way Functions — e*Gate 3.6 and 4.5.3	54
Table 9	MQSeries e*Way Functions—e*Gate 3.6.2 and 4.5.3	60
Table 10	Node Types Supporting Promotions	72
Table 11	Nodes Generated as Output	73
Table 12	Non-supported Monk Functions in e*Gate 4.5.3	77

Chapter 1

Introduction

This chapter introduces you to this guide, its general purpose and scope, and its organization. It also provides sources of related documentation and information.

1.1 Document Purpose and Scope

This guide explains how to upgrade from the SeeBeyond Technology Corporation™ (SeeBeyond™) e*Gate™ (formerly DataGate™) Integrator release 3.6 to an e*Gate Integrator release 4.5.3 environment. This explanation includes:

- Upgrade overview and procedures
- Testing and troubleshooting
- Post-upgrade maintenance
- Convenient checklist
- Sample configuration files

These chapters discuss the available SeeBeyond upgrade components, upgrade techniques and methodology, as well as testing/maintenance strategies to ensure a stable production system.

Important: *Any operation explanations given here are generic, for reference purposes only, and do not necessarily address the specifics of setting up and/or operating individual DataGate/e*Gate systems.*

This document does not contain information on how to install and/or use e*Gate. For information on these and related topics, see **"Supporting Documents" on page 12**.

1.2 Intended Audience

This guide is written for system administrators, business analysts, and information technologists who are responsible for implementing the e*Gate upgrades described previously. Experience with e*Gate 3.6 (DataGate) and training with e*Gate 4.5.3 and Monk 4.5.3 is recommended.

Note: *For more information on the Monk programming language, see the* ***Monk Developer's Reference****.*

1.3 Organization of Information

This document is organized topically as follows:

- **Chapter 1 "Introduction" g**ives a general preview of this document, its purpose, scope, and organization.
- **Chapter 2 "Upgrade Overview"** provides an overview of the general upgrade processes and operations.
- **Chapter 3 "Upgrade Procedure"** explains each step in the upgrade operation plus how to perform each step.
- **Chapter 4 "Testing and Troubleshooting"** gives you tips and procedures for testing your system before putting it into production, as well as troubleshooting techniques if any problems arise.
- **Chapter 5 "Frequently Asked Questions"** answers the most important questions you may have about your upgrade operation, including helpful hints, tips, and best practices.

In addition, there are also the following appendixes:

- **Appendix B "Upgrading Monk e*Gate 3.6 to e*Gate 4.5.3"** explains the changes in the Monk programming language as it translates from e*Gate 3.6 to e*Gate 4.5.3.
- **Appendix A "Upgrade Operation Checklist"** gives you a quick-reference, handy set of guidelines for stepping through your upgrade operation.
- **Appendix C "Moving Encapsulated Routes to Native"** explains how to move an encapsulated Route to a native Route.
- **Appendix D "Sample Configuration Files"** provides examples of the content of upgrade-related e*Way configuration files.

There is also a **Glossary** on page 129 to help you with the e*Gate 4.5.3 system's related terminology.

1.4 Writing Conventions

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

Hypertext Links

When you are using this guide online, cross-references are also hypertext links and appear in **blue text** as shown below. Click the **blue text** to jump to the section.

> For information on these and related topics, see **"Parameter, Function, and Command Names" on page 11**.

Command Line

Text to be typed at the command line is displayed in a special font as shown below.

```
java -jar ValidationBuilder.jar
```

Variables within a command line are set in the same font and bold italic as shown below.

```
stcregutil -rh host-name -rs schema-name -un user-name
-up password -ef output-directory
```

Code and Samples

Computer code and samples (including printouts) on a separate line or lines are set in Courier as shown below.

```
Configuration for BOB_Promotion
```

However, when these elements (or portions of them) or variables representing several possible elements appear within ordinary text, they are set in *italics* as shown below.

path and *file-name* are the path and file name specified as arguments to **-fr** in the **stcregutil** command line.

Notes and Cautions

Points of particular interest or significance to the reader are introduced with ***Note***, ***Caution***, or ***Important***, and the text is displayed in *italics*, for example:

Note: *The Actions menu is only available when a Properties window is displayed.*

User Input

The names of items in the user interface such as icons or buttons that you click or select appear in **bold** as shown below.

Click **Apply** to save, or **OK** to save and close.

File Names and Paths

When names of files are given in the text, they appear in **bold** as shown below.

Use a text editor to open the **ValidationBuilder.properties** file.

When file paths and drive designations are used, with or without the file name, they appear in **bold** as shown below.

In the **Open** field, type **D:\setup\setup.exe** where **D:** is your CD-ROM drive.

Parameter, Function, and Command Names

When names of parameters, functions, and commands are given in the body of the text, they appear in **bold** as follows:

The default parameter **localhost** is normally only used for testing.

The Monk function **iq-put** places an Event into an IQ.

You can use the **stccb** utility to start the Control Broker.

Additional Conventions

This guide uses the term "Windows" to refer to Windows NT, Windows 2000, and any other current Microsoft Windows operating systems.

1.5 Supporting Documents

The following SeeBeyond documents provide additional information about the e*Gate system as it is explained in this guide:

- *Creating an End-to-End Scenario with e*Gate Integrator*
- *e*Gate Integrator Alert Agent User's Guide*
- *e*Gate Integrator Alert and Log File Reference Guide*
- *e*Gate Integrator Collaboration Services Reference Guide*
- *e*Gate Integrator Installation Guide*
- *e*Gate Integrator Intelligent Queue Services Reference Guide*
- *e*Gate Integrator SNMP Agent User's Guide*
- *e*Gate Integrator System Administration and Operations Guide*
- *e*Gate Integrator User's Guide*
- *e*Insight Business Process Manager Implementation Guide*
- *e*Insight Business Process Manager User's Guide*
- *e*Xchange Partner Manager Implementation Guide*
- *e*Xchange Partner Manager User's Guide*
- *Monk Developer's Reference*
- *SeeBeyond eBusiness Integration Suite Deployment Guide*
- *SeeBeyond eBusiness Integration Suite Primer*
- *SeeBeyond JMS Intelligent Queue User's Guide*
- *Standard e*Way Intelligent Adapters User's Guide*

See the *SeeBeyond eBusiness Integration Suite Primer* for a complete list of SeeBeyond eBI Suite-related documentation. You can also refer to the appropriate Microsoft Windows or UNIX documents, if necessary.

Note: *For information on how to use a specific add-on product (for example, an e*Way™ Intelligent Adapter or IQ™), see the user's guide for that product.*

1.6 SeeBeyond Web Site

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

http://www.SeeBeyond.com

Chapter 2

Upgrade Overview

This chapter gives you a general overview of the total upgrade process in converting from e*Gate 3.6 to an e*Gate 4.5.3 environment.

2.1 Upgrading to e*Gate: Introduction

SeeBeyond provides e*Gate 3.6 customers an upgrade methodology and tool set to facilitate the move from the e*Gate 3.6 to the e*Gate 4.5.3 environment. This upgrade requires minimal effort and has minimal effect on your production environment.

The e*Gate Compatibility Components explained in this guide provide an upgrade methodology that allows you to maintain your e*Gate 3.6 routings, translations, and messages. At the same time, you can take advantage of the new applications, operating systems, standards, databases, and SeeBeyond products supported by e*Gate 4.5.3.

Important: *Before beginning your upgrade, we recommend that you read* **Chapter 5** *for answers to frequently-asked questions.*

2.2 System Requirements

The e*Gate Compatibility Components (your upgrade tools) automatically install with release 4.5.1 or later of the e*Gate Participating Host. Therefore, your system must meet or exceed the system requirements specified in the *e*Gate Integrator Installation Guide* (see this guide for details). Essentially, you need to have:

- An e*Gate Participating Host, release 4.5.1 or later
- TCP/IP connection to an FTP Server
- Additional free disk space, depending on needs

Note: *The* ***e*Gate Integrator Installation Guide*** *specifies the disk space required for a complete installation of e*Gate. Additional disk space is required to process and queue the data that the Communication Client Proxy e*Way processes. The amount necessary varies based on the type and size of the data being processed and any external applications performing the processing.*

The client components of e*Gate 3.6 have their own requirements. See that system's documentation for details.

2.3 Installation

The e*Gate Compatibility Components automatically install with the e*Gate Participating Host. See the *e*Gate Integrator Installation Guide* for information on installing the e*Gate Participating Host.

2.4 Terminology

Those who are migrating to e*Gate 4.5.3 may be accustomed to a different terminology. For clarity, this document uses the equivalent terminology shown in the following table.

Table 1 DataGate to e*Gate Terminology

DataGate term	Equivalent e*Gate 4.5.3 term
DataGate 3.6, 3.6.x	e*Gate 3.6, 3.6.x
Communication Client	Custom e*Way
Super Client, DataGateWay	Standard 3.6 e*Way
Xlate	Collaboration Rules (component)
ID	Collaboration ID
Table	Schema
Queue	Intelligent Queue (IQ)
Message	Event
Message Structure	Event Type Definition (ETD)
Routing Paradigm	Publication/subscription (pub/sub) paradigm
Data translation	Data transformation
Feeder and eater DataGateWays	**stcewfile** Standard e*Way (inbound and outbound)
dgcmd	**stccmd**

For more information on e*Gate terms, see the **"Glossary" on page 129**.

2.5 Supported Versions

e*Gate 3.6, 3.6.1, and 3.6.2 are the releases supported for the upgrade process explained in this guide. Supported release 3.6 configuration formats include:

- Only "Tables" mode scripts for IDs & Xlates
- Only "Monk" mode scripts for IDs & Xlates
- A mixture of "Monk" and "Table" mode scripts for IDs & Xlates

Note: *Any 2.x Route Tables that you are running in your 3.x environment are also supported.*

Refer to **"Supported e*Gate 3.6 e*Ways for Configuration File Conversion" on page 24** for a complete list of e*Ways that can utilize the **dgw2ew** command utility.

2.6 e*Gate Compatibility Components

The e*Gate Compatibility Components automate the upgrade process from e*Gate 3.6 to e*Gate 4.5.3. This tool set consists of the following components:

- **Schema Configuration Utility (stcdgschema)** allows you to create an e*Gate 4.5.3 schema from e*Gate 3.6 Route Table files. This command configures a schema to use:
  - **Communication Client Proxy e*Way,** which enables both standard and custom e*Gate 3.6 e*Ways to exchange data with e*Gate 4.5.3 IQ components
  - **Route Table Collaboration (RTC) Service**, configured in the inbound Communication Client Proxy e*Way of the schema created by the **stcdgschema** command to transport and manipulate data in accordance with the e*Gate 3.6 Route Table

Note: *The Communication Client Proxy e*Way and RTC Service are installed with the Participating Host and can be used even if the Schema Configuration Utility is not used.*

- **e*Way Configuration File Conversion Utility** uses the original e*Gate 3.6 e*Way configuration file to help generate an equivalent e*Gate 4.5.3 e*Way's configuration files. Use this tool when substituting e*Gate 3.6/Proxy e*Way combinations with the functional equivalent of e*Gate 4.5.3 e*Ways.

This section explains the operation and use of each of these components in detail.

2.6.1 Schema Configuration Utility

This section explains the Schema Configuration Utility tool (**stcdgschema.exe**) in the Upgrade tool set, as well as the basics of how this tool operates.

See **"Running the stcdgschema Utility" on page 32** for an explanation of the procedure detailing how to use the **stcdgschema** command. This command offers you the options shown in the following table.

Table 2 Option Arguments for stcdgschema Command

Command argument option	Function
-rh	Name of the e*Gate 4.5.3 Registry Host.
-rs	Name of the new e*Gate 4.5.3 schema to be created.
-o	Name where the output files are to be placed.
-i	Full path and name of the **.dgt** file.
-dg	Full path name of the directory defined by the $DATAGATE environment variable.
-dgenv	Full path name of the directory defined by the $DATAGATE_ENV environment variable.
-egateclient	Full path name of the e*Gate 4.5.3 client subdirectory.
-noJavaConnectionPoints	Creates a schema that does not use Java Connection Points.
-q [1-500] (optional)	Number of IQ Managers to define within this schema (optional). The number of IQ Managers must be equal to or less than the number of IQs. The default is 1 and the maximum is 500.
-singleroute (optional)	Generates a single route using a Business Object Broker (BOB). See **"Associating a BOB with a Route Table" on page 21**. ▪ Using this flag means all routing takes place in a BOB, which runs the RTC (the BOB picks up the entire Route table). ▪ Not using this flag means the Route table is broken up by inbound instances, and inbound e*Ways all run the RTC (the e*Ways only pick up the tables that are related to them). New copies of the table files are generated based on the inbound routes for each DataGateWay and are stored in the original table file's subdirectory.
-loop (optional)	Include the route that loops into itself.

Important: *If you do not use the* ***-singleroute*** *flag, you must ensure that there are no blank lines in any of the table files and that all comments start in column 1 of the file.*

For a more detailed explanation of these command arguments, see step 4 in the **procedure on page 32**.

The Schema Configuration Utility automatically creates an e*Gate 4.5.3 schema from e*Gate 3.6 Table files. This operation takes place as a series of processes configuring the following components:

- Proxy e*Ways
- RTC Service and pub/sub definitions
- IQs
- Event Types

The rest of this section provides an overview of these processes.

Creating Proxy e*Ways

The Schema Configuration Utility (**stcdgschema.exe**) configures Proxy e*Ways as follows:

- From the e*Gate 3.6 Port Table, **stcdgschema** extracts the logical name of each e*Gate 3.6 (custom or standard) e*Way.
- It compares that list to the e*Gate 3.6 Route Table.
- The utility identifies each e*Gate 3.6 e*Way as either an inbound or outbound e*Way.
- Finally, the inbound Proxy e*Ways are created and then configured with the RTC Service, and the outbound Proxy e*Ways are created and configured as Pass Through.

Note: *For complete information on the Proxy e*Way, see the* ***Communication Client Proxy e*Way User's Guide****.*

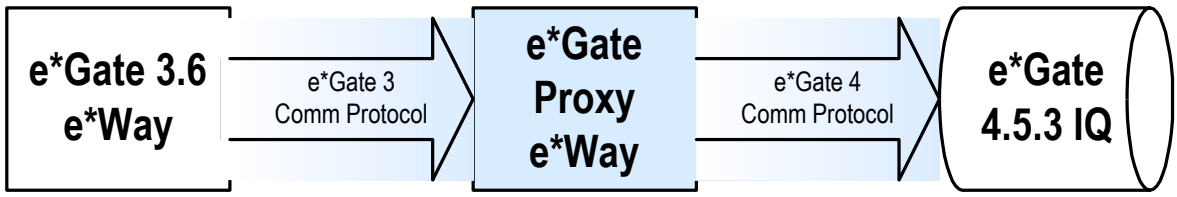
Proxy e*Way Operation

The Proxy e*Way (**stcewproxy.exe**) enables a custom Communication Client (standard e*Gate 3.6) e*Way to exchange data with e*Gate 4.5.3 via an IQ. The Proxy e*Way converts the e*Gate 3.6 communications protocol to the communications protocol used by e*Gate 4.5.3. The reverse is also true, depending on the direction of communication. Once the Proxy e*Ways establish communication with the e*Gate 3.6 e*Ways, they are able to exchange data with e*Gate 4.5.3 e*Ways.

Inbound Proxy e*Ways are configured to run the RTC Service and publish to all the IQs of the outbound Proxy e*Ways (as determined by the e*Gate 3.6 Route Table). The outbound Proxy e*Ways all use the Pass Through Collaboration Service, because all of the data transformations have already taken place in the inbound Proxy e*Ways.

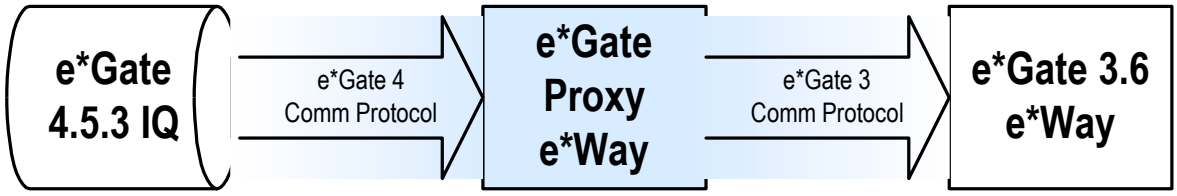
This setup allows an inbound Proxy e*Way to publish to an IQ, enabling data to be transferred between the e*Gate 3.6 e*Way, the Proxy e*Way, the IQ, and the e*Gate 4.5.3 environment (see the following figure).

Figure 1 Inbound Proxy e*Way Operation



Outbound Proxy e*Ways allow data to pass through unchanged (Pass Through Service). The following figure shows the operation of outbound Proxy e*Ways.

Figure 2 Outbound Proxy e*Way Operation



The **stcdgschema** command automatically generates the necessary **.cfg** and **.sc** files that support the configuration of the Proxy e*Way. A Proxy e*Way is then created for each e*Gate 3.6 e*Way. The e*Gate 3.6 e*Ways can interact with e*Gate 4.5.3 by using the Proxy e*Way—the e*Ways operate in combination.

Creating the Route Table Collaboration Service

The RTC Service (**stc_routetablecollab.dll**) replicates the behavior of the e*Gate 3.6 server. You can configure this service manually, or you can use the Schema Configuration Utility to do it automatically.

RTC Service Operation

After you run the **stcdgschema** command, the e*Gate 3.6 Table files become encapsulated within the new e*Gate 4.5.3 schema. This process happens via the RTC Service as it is configured within the inbound Proxy e*Ways. In turn, this encapsulated schema is completely assimilated into the e*Gate 4.5.3 environment.

The **stcdgschema** utility configures the RTC Service in the inbound Proxy e*Ways in any schema it creates. The RTC Service transports and manipulates data in accordance with the e*Gate 3.6 Route Table.

In addition, **stcdgschema** sets up the appropriate e*Gate publications and subscriptions. The new schema duplicates all the original e*Gate 3.6 publication-and-subscription (pub/sub) definitions.

After the **stcdgschema** utility has created a schema, the RTC Service performs the following operations in the schema's e*Gate 4.5.3 environment:

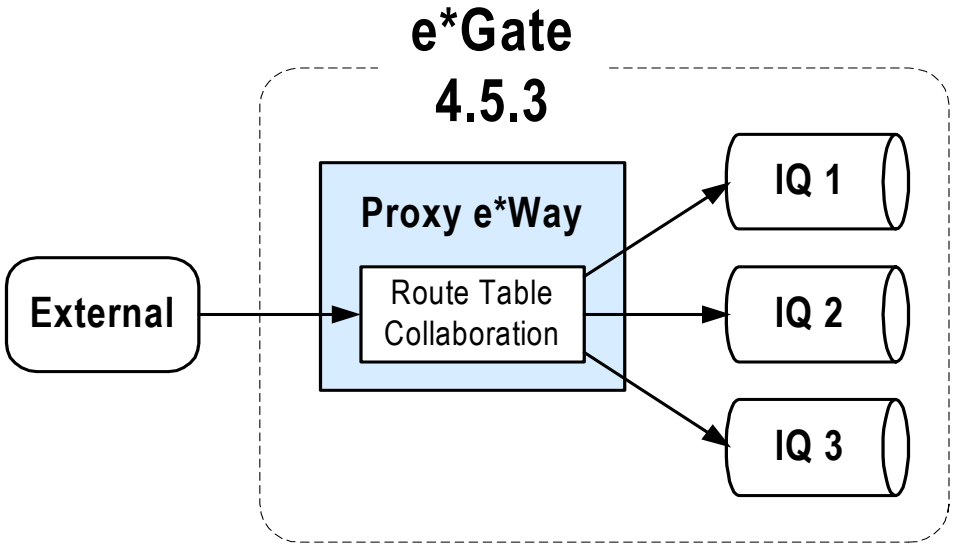
- Reads the e*Gate 3.6 Route Table files, which contain port, ID, routing, and data transformation information.
- Uses the pub/sub rules created by the **stcdgschema** tool to read incoming data from the IQ then performs the identifications, data transformations, and routings defined in the e*Gate 3.6 environment.
- Publishes to an IQ via a BOB or an e*Way; only one instance of the RTC Service per any BOB or e*Way is allowed.

Creating Event Types and Intelligent Queues

In addition to Proxy e*Ways, the RTC Service, and pub/sub definitions, the Schema Configuration Utility creates the following components:

- **Event Types** utilized in e*Gate 4.5.3 are created from the logical entries in the e*Gate 3.6 Port Table.
- **IQ Managers**, one or more are created, as desired (up to 500).
- **One IQ** for each outbound e*Gate 4.5.3 e*Way; see the following figure.

Figure 3 Creation of IQs



Note: *For a bidirectional e*Gate 3.6 e*Way, two IQs are created, one for each direction.*

IQ Managers

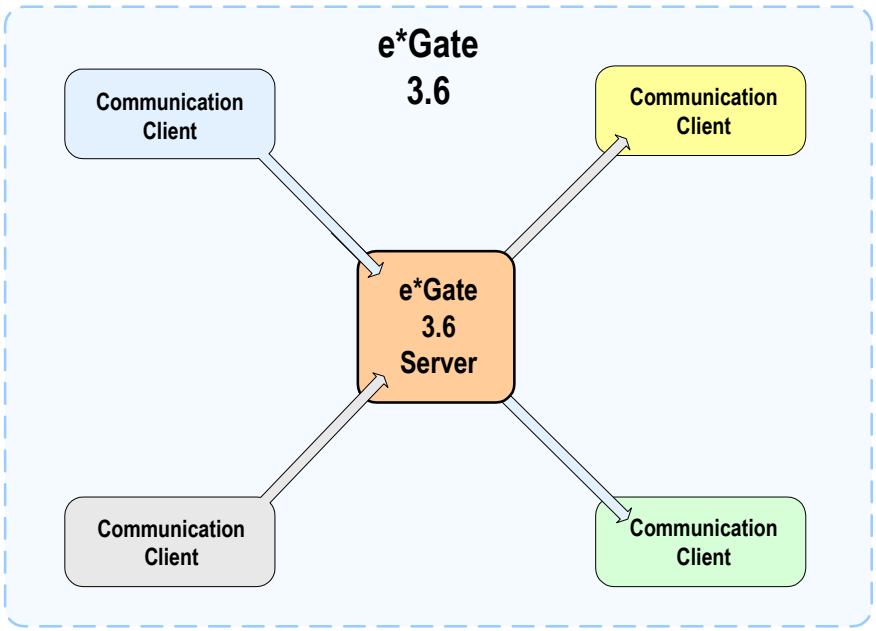
By default, only one IQ Manager is created. If your schema contains a large number of IQs, (using the **-q** command option) you can create up to 500 IQ Managers. The number of IQs is then divided by the number of IQ Managers specified, and the resulting number of IQs is assigned to each IQ Manager. The number of IQ Managers does not have to be equally divisible by the number of IQs, but be sure to create a number of IQ Managers equal to or less than the number of IQs.

Note: *For high-volume routes, SeeBeyond recommends no more than three IQs per one IQ Manager.*

Associating a BOB with a Route Table

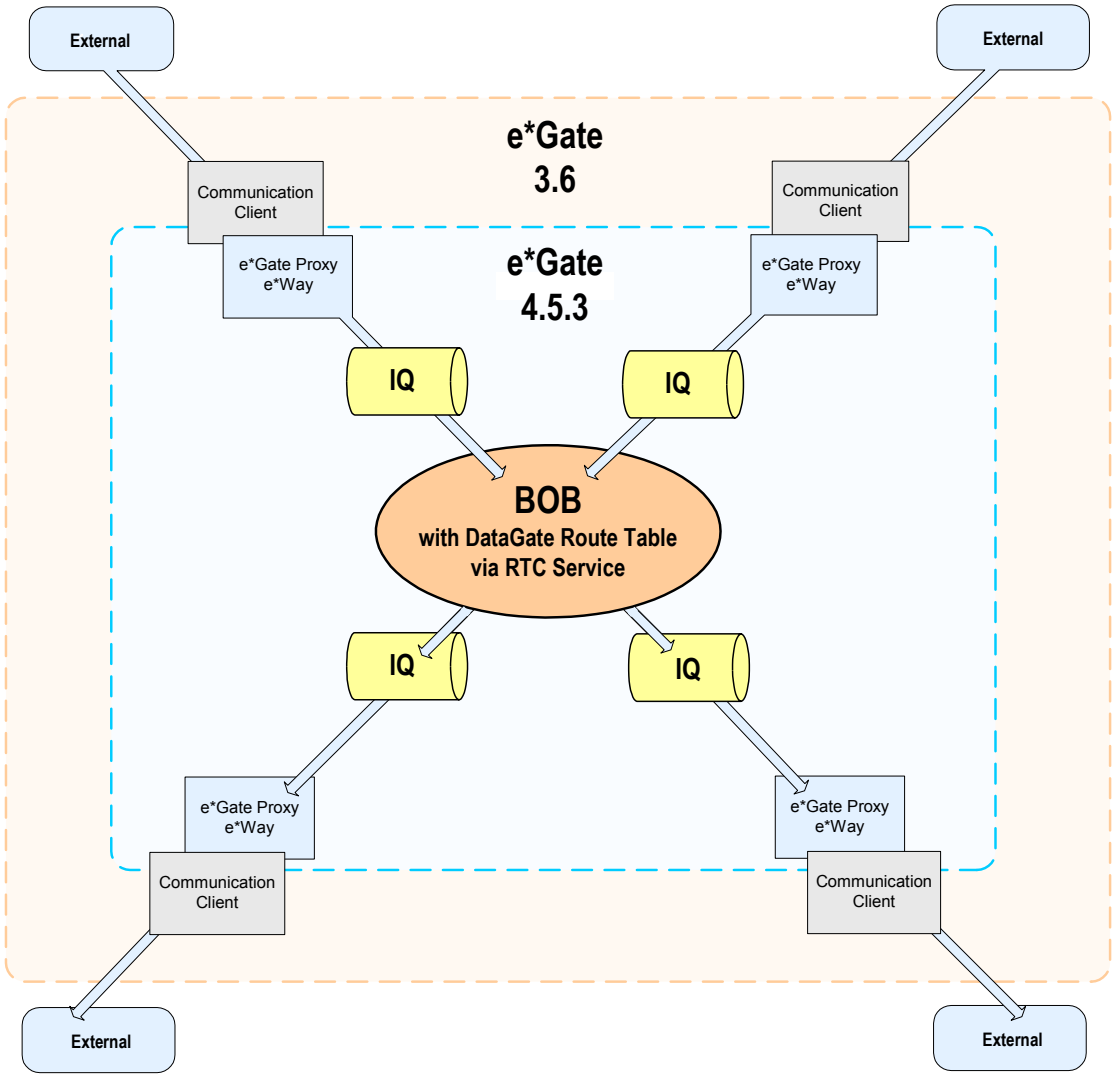
Use of the **-singleroute** flag (see **Table 2 on page 17**) with **stcdgschema** takes an e*Gate 3.6 Route Table and configures it in relation to a single BOB in e*Gate 4.5.3. For example, if you start out with the e*Gate 3.6 configuration shown in Figure 4 then use the **-singleroute** flag**,** the result in e*Gate 4.5.3 looks like the configuration shown in Figure 5.

Figure 4 Beginning with Routes in e*Gate 3.6



Use of the **-singleroute** flag with **stcdgschema** causes the e*Gate 3.6 Route Table information to be stored with the resulting BOB's Collaboration via the RTC Service. In e*Gate 4.5.3, each of the encapsulated (e*Gate 3.6) Communication Clients communicates with the BOB via an e*Gate 3.6/Proxy e*Way combination and an e*Gate 4.5.3 IQ (see Figure 5).

Figure 5 Associates Route with Single BOB in e*Gate 4.5.3



Note: The incoming IQ sends an acknowledged message back to the External before anything happens. This message does not confirm any of the actions that are still to come.

The use of this **-singleroute** feature is optional. If it is not used, the RTC Service is configured for all of the inbound Proxy e*Way Collaborations and Pass Through for all of the outbound.

If you do use the **-singleroute** flag, the **stcdgschema** utility configures Pass Through as the Collaboration Service for all of the Proxy e*Ways, inbound and outbound. In this case, only the Collaboration for the BOB is configured with the RTC Service.

2.6.2 e*Way Configuration File Conversion Utility

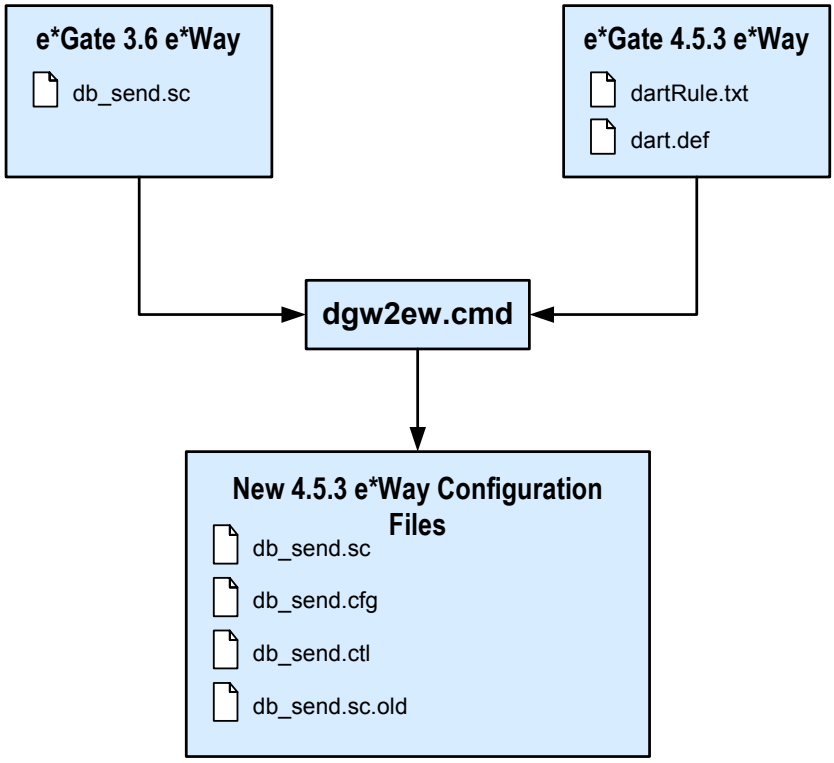
Use this utility when you want to replace an e*Gate 3.6/Proxy e*Way combination with an equivalent e*Gate 4.5.3 e*Way. The e*Way Configuration File Conversion Utility (**dgw2ew.cmd**), available under Windows, uses the original e*Gate 3.6 e*Way configuration file (**.sc**) to help generate an equivalent e*Gate 4.5.3 e*Way's configuration files (**.sc** and **.cfg** files).

Utility Operation

The e*Way Configuration File Conversion Utility reads the e*Gate 3.6 e*Way configuration file (**.sc**), uses the e*Gate 4.5.3 e*Way default definition file (**.def**) as a template, and applies the rules contained in the e*Gate 3.6 e*Way Rules file (**.txt**) to the parameters in the **.sc** file.

It then generates new **.sc** and **.cfg** files that can be used in an e*Gate 4.5.3 e*Way. Additionally, **dgw2ew.cmd** creates a control file (**<*e*Way*>.ctl**) that simplifies the process of committing those files to the e*Gate 4.5.3 Registry. See **Figure 6 on page 23** for details.

Figure 6 e*Way Configuration File Conversion Operation



Examples of the files shown in the previous figure can be found in**"Sample Configuration Files" on page 81**.

Once the conversion is complete, the e*Way is ready to interact directly with e*Gate 4.5.3, eliminating the need to use the original e*Gate 3.6/Proxy e*Way combination.

Supported e*Gate 3.6 e*Ways for Configuration File Conversion

This utility can be used with these standard e*Gate 3.6 e*Ways that have corresponding e*Gate 4.5.3 e*Ways:

- Batch
- COM/DCOM
- CORBA Visibroker
- DART for ODBC
- DART for Oracle
- DART for Sybase
- FTP
- MQSeries
- PeopleSoft Message Agent
- SAP ALE
- SAP BAPI
- SAP BDC
- SAP EDI
- Siebel EIM
- Siebel Event Driven
- TCP/IP HL7

The procedure detailing how to use **dgw2ew.cmd** can be found in **"Running dgw2ew.cmd" on page 40**.

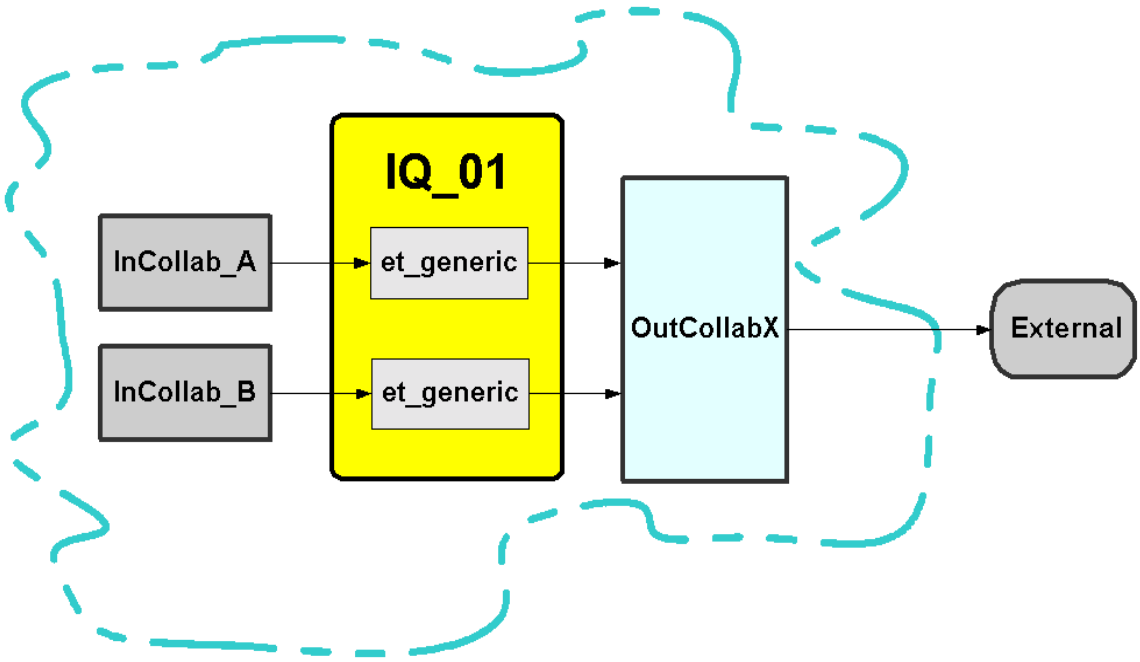
2.6.3 FIFO

FIFO stands for *first in first out*. This means that the first Event that reaches an IQ will be the first Event to leave the IQ. This is not always the case. What follows is an example that demonstrates why "first in" is not always "first out," followed by an example of how FIFO functions.

Outbound e*Way with a Collaboration subscribing to multiple inbound Collaborations

Even though the messages in this example are stored in FIFO order in the IQ, the messages do not always appear in FIFO order after the outbound Collaboration picks up the messages and processes them. This happens because the outbound Collaboration subscribes to multiple Event Type/Publisher pairs. The inbound Collaboration processes each of its subscriptions in turn. See **"Example of Collaboration Subscribing to Event Type Publisher Pairs" on page 25**.

Figure 7 Example of Collaboration Subscribing to Event Type Publisher Pairs



Since **OutCollabX** is subscribing to Event Type/Publisher pairs, it processes the messages in turn, one from **InCollabA**, one from **InCollabB**, one from **InCollabA**, and so on.

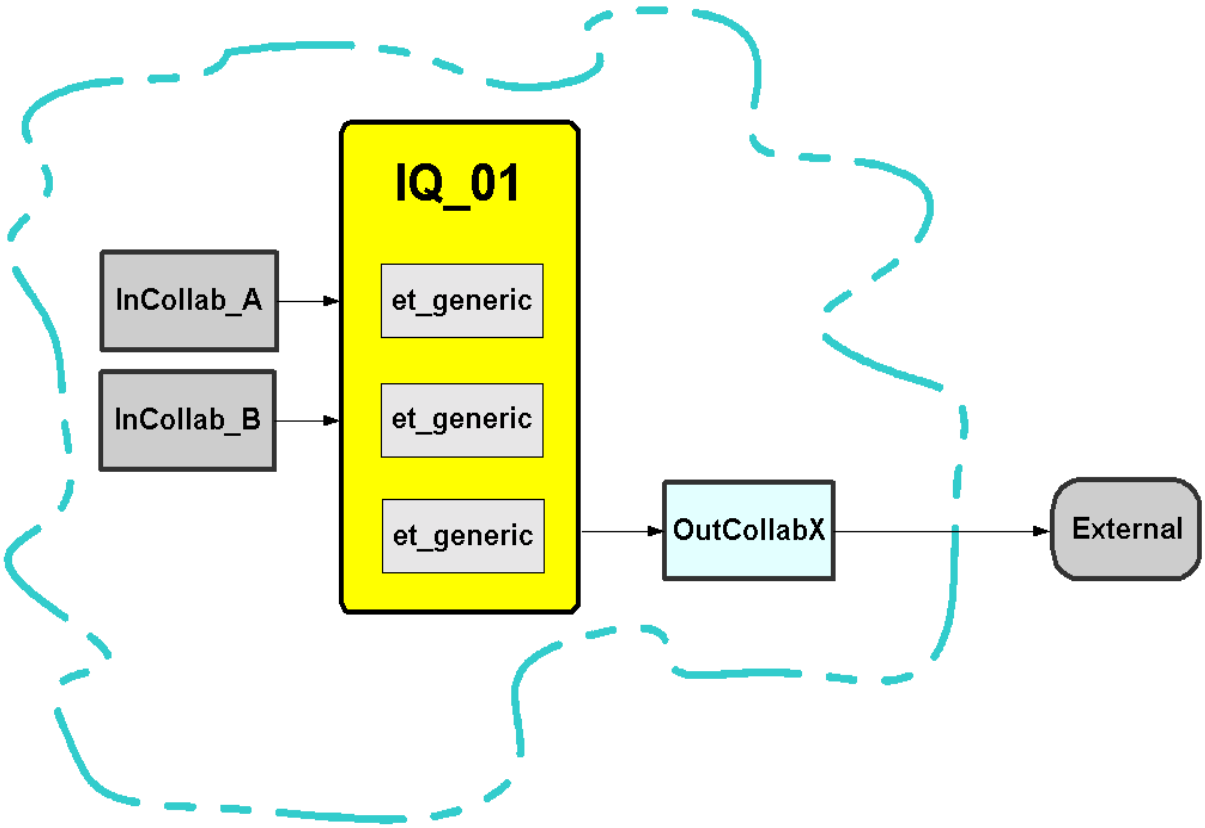
Enqueued order is the key. If for some reason **OutCollabX** goes down, the order in which it subscribes to the Event Type/Publisher pairs may not match the order in which they entered the IQ when it is again up and running.

Note: *This is default behavior of the queues.*

Insuring that FIFO is always *first in first out*

To avoid the problem demonstrated in the above example, subscribe to the IQ instead of the publishing Collaborations. You can do this by using a JMS e*Way Connection. See **"Example of Collaboration Subscribing to an IQ" on page 26**.

Figure 8 Example of Collaboration Subscribing to an IQ



Since **OutCollabX** is subscribing to the Event Type instead of the Event Type/Publisher pairs, the messages are pulled in the order in which they were put on the IQ.

BOBs and the Standard IQ FIFO

BOBs do not have access to FIFO when they use **stcbob.exe**. However, you can give a BOB access to FIFO by changing its executable to **stceway.exe**. This gives it a configuration file and basically turns it into a Multi-Mode e*Way. The configuration file for the Multi-Mode e*Way contains the switch that turns on the capability to use the "Standard IQ FIFO." The default setting is **No**. To use FIFO, this setting must be changed to **Yes**.

To turn a BOB into a Multi-Mode e*Way

1. In the Enterprise Manager, right-click a BOB.
2. On the shortcut menu, click **Properties**. The **Business Object Broker - <*BOB name*> Properties** dialog box appears.
3. Under **Executable file**, click **Find**.
4. In the **File Selection** dialog box, select **stceway.exe** and click **Select**. The **Configuration file** area of the BOB (which is now a Multi-Mode e*Way) becomes active, containing a configuration file that has already been created.

To change the setting of the Standard IQ FIFO

1. Under **Configuration file** on the Multi-Mode e*Way, click **Edit**. The **Edit Settings for <*directory/configuration file name*>** dialog box appears.
2. In the **Goto Section**, select **General Settings**.
3. Under **Standard IQ FIFO**, select **YES**.
4. On the **File** menu, select **Save**.
5. On the **Save As** dialog box that appears, enter a name for the configuration file and click **Save**.

For information on FIFO, see the *e*Gate Integrator Intelligent Queue Services Reference Guide*; for information about Multi-Mode e*Ways, see the *e*Gate Integrator User's Guide*.

When FIFO is activated

Setting the FIFO switch to **Yes** in the **stceway.exe** allows the Multi-Mode e*Way to check for a triggering Standard IQ message, obtaining one message for every triggering Event Type published by every publisher to which the e*Way subscribes. After obtaining the messages, the e*Way then compares message priorities to find the message or messages with the highest priority (that is, the lowest priority number). The e*Way then searches for the one message with the earliest enqueue time among the messages with the highest priority. When a message is identified that meets this criteria, it is delivered. The remaining messages (if any) are then cached by the e*Way for use when the process repeats. When the process begins again, the e*Way will not attempt to obtain a new triggering Standard IQ message for a particular Event Type published by a particular publisher if there is one or more messages still in the cache.

Note: *Even though triggering Event Types must be published to Standard IQs, they can be published to multiple IQs that are controlled by multiple IQ Managers.*

Generating a Multi-Mode e*Way with FIFO

To generate a Multi-Mode e*Way with FIFO capability, use the **-noJavaConnectionPoints** flag when executing the **stcdgschema** command. For information about **stcdgschema**, see **"Schema Configuration Utility" on page 17**.

2.7 Upgrade Methodology

The e*Gate 3.6-to-4.5.3 upgrade methodology involves the following basic steps:

- Using the following e*Gate Compatibility Component tools:
  - Schema Configuration Utility (**stcdgschema** command)
  - RTC Service
  - Proxy e*Way
- Substituting the Proxy e*Ways with equivalent e*Gate 4.5.3 e*Ways (where necessary and/or desired)

This section summarizes the general upgrade methodology.

2.7.1 Using e*Gate Compatibility Component Tools

These utilities are the essential tools of the upgrade process. The previous section, **"e*Gate Compatibility Components" on page 16**, described these tools in detail. Use the following basic steps to implement these tools during your upgrade operation:

1. In the test environment, run **stcdgschema** for each e*Gate 3.6 environment being upgraded.
2. Perform any and all necessary tests to ensure the correct operation of converted/upgraded components in the test environment.
3. Move the test environment to production.
4. Perform any and all necessary tests to ensure the correct operation of converted/upgraded components in the production environment.

See **Chapter 3** for a complete explanation of this procedure.

2.7.2 Substituting Equivalent e*Gate 4.5.3 e*Ways

After you upgrade, you may want to convert some or all of your Proxy e*Ways to equivalent e*Gate 4.5.3 e*Ways for one of the following reasons:

- You want to take advantage of the latest features.
- You need support for the latest third-party applications and platforms (for example, Oracle 9i).

The major advantage of using equivalent e*Gate 4.5.3 e*Ways is that you are taking complete advantage of the features and functions associated with the e*Gate 4.5.3 architecture.

Convert any e*Gate 3.6 e*Way configurations to e*Gate 4.5.3 e*Ways using the **dgw2ew.cmd** procedure (see **"Running dgw2ew.cmd" on page 40**).

Important: *SeeBeyond recommends that you take this step before moving from the test to the production environment.*

Note: *Using equivalent e*Gate 4.5.3 e*Ways could require some preparatory work before you start to upgrade, depending upon your use of custom Monk functions. This preparatory work generally requires the moving of Monk functions, verifying the current function names and/or behavior and the files using those functions.*

2.7.3 Using Proxy e*Ways

You may want to continue using the upgraded Proxy e*Ways for either of the following reasons:

- You have custom e*Ways.
- You do not wish to upgrade the 3.6 e*Way to the e*Gate 4.5.3 equivalent.

You can continue using Proxy e*Ways if you want to upgrade as quickly as possible. A major advantage of Proxy e*Ways is that they require little preparatory work before the upgrade. This process is the simplest to implement.

Any new e*Ways you need can be added by adding a new e*Gate 4.5.3 e*Way. As time and resources permit, you can convert any desired e*Gate 3.6 e*Way configurations, translations, and messages into the e*Gate 4.5.3 format, to become totally operational in the e*Gate 4.5.3 environment.

Keep in mind that Proxy e*Ways take the least advantage of e*Gate 4.5.3 features and functions. Additionally, any changes to routing information made after the upgrade must be completed in both the e*Gate 3.6 and e*Gate 4.5.3 environments.

Note: *After converting an e*Gate 3.6 table that uses the **rpc-external** function to call an external program, you must first ensure that the process **ext_proc** is running in the e*Gate 3.6 environment. This process is normally invoked by the e*Gate 3.6 server in the e*Gate 3.6 run-time environment. In these cases, do **not** run the newly converted e*Gate 4.5.3 schema until you are sure this process is running. The Proxy e*Ways themselves do not start the **ext_proc** process.*

Chapter 3

Upgrade Procedure

This chapter explains the steps you take to upgrade from e*Gate 3.6 to the e*Gate 4.5.3 environment.

3.1 Upgrading to e*Gate 4.5.3: Overview

The upgrade process has the following basic components:

- Ensure the prerequisite procedures are completed before using any of the Compatibility Components
- Follow each step to complete your upgrade, using the **stcdgschema** utility command then commit the files to the e*Gate Registry

In addition, you may need to use the following upgrade features:

- Undefining the DGOS environment variable
- Making configuration changes to the Proxy e*Way
- Running **dgw2ew.cmd** to convert Proxy e*Ways to e*Gate 4.5.3 e*Ways

Appendix A provides a checklist. This chapter explains all of these operations in detail.

3.2 Upgrade Prerequisites

Before implementing the upgrade from the e*Gate 3.6 to an e*Gate 4.5.3 environment, complete the following steps:

1. Be sure you have the required hardware necessary for e*Gate 4.5.3 as advised in the *e*Gate Integrator Installation Guide* and the *SeeBeyond eBusiness Integration Suite Deployment Guide*.
2. Upgrade the necessary third-party software and operating systems as needed, as outlined in the *e*Gate Integrator Installation Guide*.
3. Install e*Gate 4.5.3. This includes all applicable e*Way add-ons. See the *e*Gate Integrator Installation Guide* for details.

4 If you are using any of the UNIX platforms, source the **egateclient.sh** or **egateclient.csh** file (as appropriate for your login shell) in the **client** subdirectory of the e*Gate directory tree.

5 Test your e*Gate 4.5.3 environment by creating the scenario documented in *Creating an End-to-End Scenario with e*Gate Integrator.*

6 Set your environment variables (such as **$DATAGATE** and **$DATAGATE_ENV**), your PATH, and your Shared Library PATH.

Note: *See your operating system for the name of your Shared Library Path on UNIX.*

7 Test your e*Gate 3.6 environment and verify that your existing e*Gate 3.6 e*Ways are operational.

8 Identify your e*Gate 3.6 e*Ways as either Proxy or e*Gate 4.5.3 e*Way Configuration File Conversion candidates. See **"Supported e*Gate 3.6 e*Ways for Configuration File Conversion" on page 24** for a complete list.

9 If you have standard e*Gate 3.6 e*Ways and are going to use the e*Gate 4.5.3 e*Way Configuration File Conversion utility, identify all Monk scripts used in your e*Gate 3.6 e*Ways. All scripts must be compatible with Monk version 4.5.3 standards (see **Chapter 4** for more information). Edit the scripts as needed.

10 Set up a test environment to run the upgrade.

Note: *For more information on the Monk programming language, see the* ***Monk Developer's Reference****.*

3.3 Upgrade Procedure

The following steps provide a procedural overview of the upgrade operation:

1 Complete all the prerequisite procedures described under **"Upgrade Prerequisites" on page 30**.

2 Run **stcdgschema** for each table being upgraded and commit any new schemas to the e*Gate 4.5.3 Registry.

3 Confirm that the schema(s) and Proxy e*Ways were created correctly.

4 Test all Proxy e*Ways with data. This includes Proxy e*Ways that were created for e*Gate 3.6 e*Ways that will convert to e*Gate 4.5.3 e*Ways.

5 Convert any desired e*Gate 3.6 e*Way configurations to e*Gate 4.5.3 e*Ways using **dgw2ew.cmd** procedure.

6 Test the new e*Gate 4.5.3 e*Ways with data.

7 Move the test environment to production.

The rest of this section explains the basic upgrade procedure in detail.

3.3.1 Running the stcdgschema Utility

This procedure creates an e*Gate 4.5.3 schema from the original e*Gate 3.6 Table Files, including:

- Communication Client Proxy e*Ways (inbound using the Route Table Collaboration Service)
- At least one IQ Manager or as many as 500
- An IQ for each outbound e*Way

Note: *Throughout these instructions, you are prompted for a user name, password, Registry Host name, and schema name. Use the same values throughout these instructions. The default e*Gate Administrator user name and password are listed in the* ***Readme.txt*** *file on the e*Gate installation CD-ROM disc 1.*

To run the stcdgschema command utility

1. Create a temporary directory to place the **stcdgschema** output files (the new **.cfg**, **.sc**, **.ctl**, and **.txt** files).
2. Identify the e*Gate 3.6 Table file (**.dgt**) to convert.
3. Change (**cd**) to the temporary directory you created in step 1.
4. To convert the Table Files to a schema, from the command prompt, type:

```
stcdgschema -rh registryhost -rs schemaname -o outputdirectory -i
    inputfilename -dg DataGate_Directory -dgenv
    Datagate_Environment -egateclient client_directory -q
    number_of_IQManagers [-singleroute]
```

Where:

- ***registryhost*** is the name of the e*Gate 4.5.3 Registry Host.
- ***schemaname*** is the name of the e*Gate 4.5.3 schema that supports the migrated components. You can provide the name of your choice.
- ***outputdirectory*** is the name of the current directory you are in and where the output files are to be placed. You can optionally enter a dot (“.”) to indicate the current directory.
- ***inputfilename*** is the full path and name of the e*Gate 3.6 Table file (**.dgt**).
- ***Datagate_Directory*** is the full path name of the directory defined by the $DATAGATE environment variable.
- ***Datagate_Environment*** is the full path name of the directory defined by the $DATAGATE_ENV environment variable.
- ***client_directory*** is the full path name of the e*Gate 4.5.3 client subdirectory within the e*Gate directory tree.
- ***number_of_IQManagers*** is the number of IQ Managers to define within this schema (this parameter is optional).

- **-singleroute** (optional), if specified, indicates that all messages are to be routed through a single BOB. New copies of the table files are generated based on the inbound routes for each DataGateWay and are stored in the original table file's subdirectory.

Use the optional ***-q*** argument if you want multiple IQ Managers. After that argument, specify the desired number. The number of IQs is then divided by the number of IQ Managers specified, assigning a number of IQs to each Manager. The number of IQ Managers does not have to be equally divisible by the number of IQs, but the number of IQ Managers should be less than the number of IQs.

Important: *For IQs of high-volume usage, SeeBeyond recommends no more than three IQs per IQ Manager. If* ***-q*** *is not specified, only one IQ Manager is created.*

Note: *The upgrade tools will generate schemas with JMS IQs. By default, sync is enabled in the JMS IQ Manager.*

the SeeBeyond Standard IQ with SYNC disabled.

The **stcdgschema** command creates a number of files. Most of the files have the names of the components whose configuration data they contain. Two files with **.txt** and **.ctl** extensions are created. These files have the same name as the schema you specified on the command line. For example, if you specified the schema name **new_schema**, the files are named **new_schema.txt** and **new_schema.ctl**.

Important: *If you do not use the* ***-singleroute*** *flag, you must ensure that there are no blank lines in any of the table files and that all comments start in column 1 of the file.*

5 Import the schema into e*Gate by running the following commands:

```
stcregutil -rh $rh -rs $rs -un $un -up $up         -i $rs.txt
stcregutil -rh $rh -rs $rs -un $un -up $up -fc . -ctl $rs.ctl
```

Where:

- ***rh*** is the name of the e*Gate 4.5.3 Registry Host.
- ***rs*** is the name of the e*Gate 4.5.3 schema that supports the migrated components. You can provide the name of your choice.
- ***un*** is the user name as defined within the specified schema.
- ***up*** is the password corresponding to the user name.
- ***.txt*** and ***.ctl*** are the two files created by **stcdgschema**.

6 Open the e*Gate Enterprise Manager.

Note: *Changes are committed (saved) to the Registry as you make them. For more information on e*Gate operation and how to use the Enterprise Manager, see the* ***e*Gate Integrator User's Guide****.*

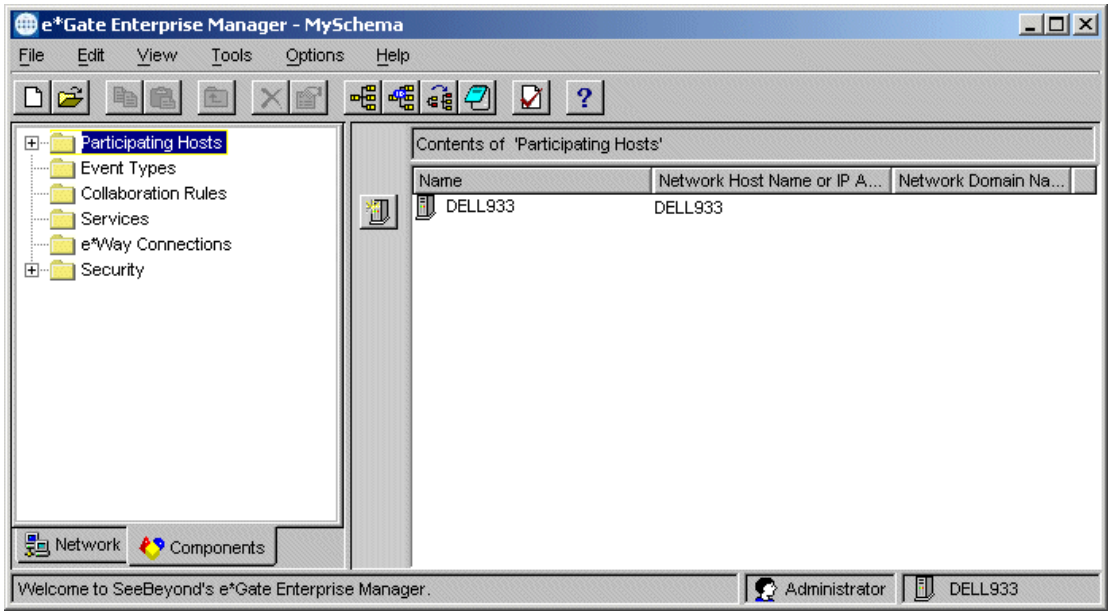
Note: *See the* ***e*Gate Integrator System Administration and Operations Guide*** *for step-by-step instructions on how to use the Import Wizard.*

7 When prompted, log in to the e*Gate 4.5.3 Registry Host.

8 The **Open a Schema** dialog box appears. Select the schema and click **OK**.

The Enterprise Manager appears, as shown in the following figure.

Figure 9 Enterprise Manager Main Window



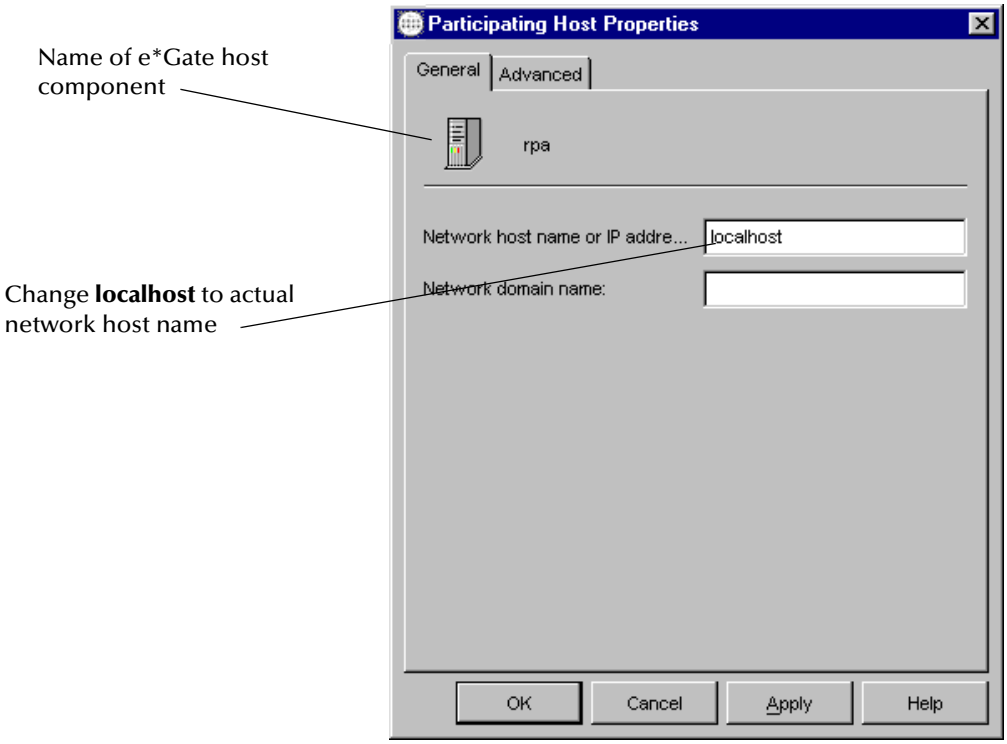
9 The left pane operates much like Windows Explorer. Click the plus sign (as shown in the previous figure) to open the **Participating Hosts** folder.

10 Right-click the host icon.

11 When the shortcut menu appears, click **Properties.**

12 Under the **General** tab, replace the **localhost** name with the host's actual network name, and click **OK** (see the following figure).

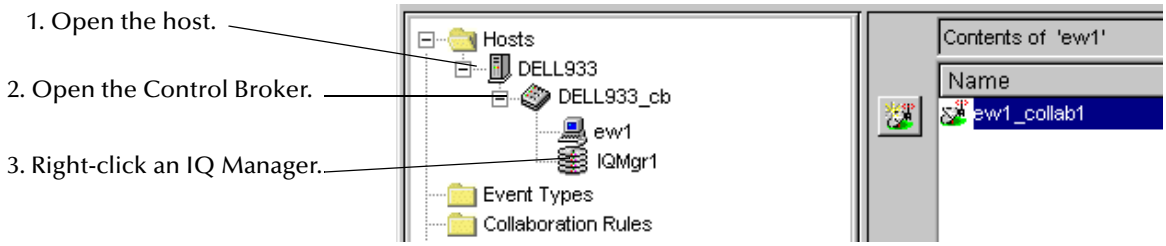
Figure 10 Participating Host Properties Dialog Box



13 If you did not use the **-q** argument to define multiple IQ Managers, skip ahead to step 15. If you defined multiple IQ Managers with the **-q** argument in step 4 in these procedures, the **stcdgschema** utility assigned unique ports for each IQ Manager. Such ports may not be available at your installation. Do the following actions for *each* IQ Manager:

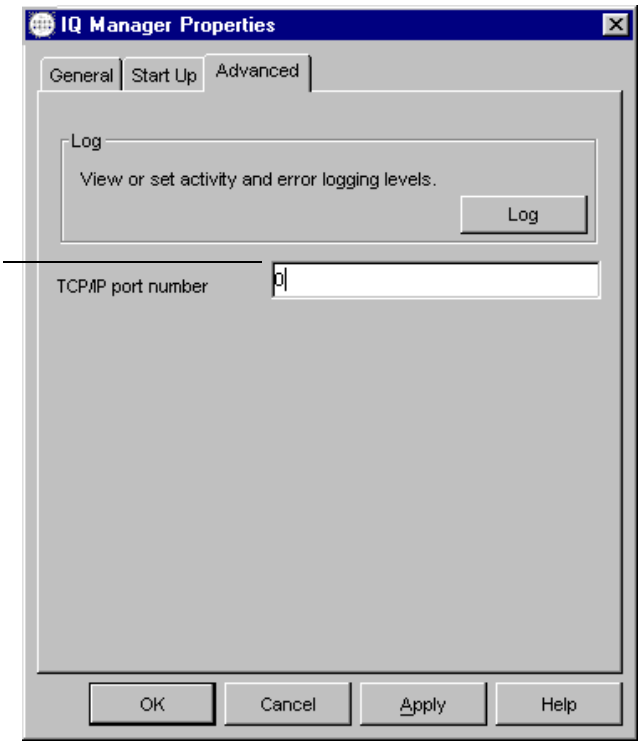
- In the left pane of the Enterprise Manager, open the host, and open the Control Broker under that host.
- Right-click an IQ Manager. When the shortcut menu appears, click **Properties.**

Figure 11 Enterprise Manager with Components Displayed



The **IQ Manager Properties** dialog box appears (see **Figure 12 on page 36**).

Figure 12 IQ Manager Properties Dialog Box



Verify the appropriate TCP/IP port number for each IQ Manager.

14 In the **IQ Manager Properties** dialog box, perform the following:

- Select the **Advanced** tab.
- Verify that the TCP/IP port assigned to this IQ Manager is available.
- Select whatever port is appropriate at your installation.
- Click **OK**.

15 If you need to convert 3.6 e*Way configuration files, see **"Running dgw2ew.cmd" on page 40** for the conversion procedure. If you have completed your upgrade process at this point, prepare to test your data in the e*Gate 4.5.3 environment.

3.3.2 Committing Files to the e*Gate Registry

This section describes the procedures that you use whenever files are committed to a schema within the e*Gate 4.5.3 Registry. This procedure is used at several points during the upgrade process, as well as whenever you want to move a file to the e*Gate Registry.

You must commit files using these procedures, rather than simply copying files to a file system directory, as it ensures that the e*Gate 4.5.3 Registry server can properly manage the files.

Within this guide, all files must be committed to the same directory: **monk_scripts/common**. This pathname is relative to the root directory of the target schema's file repository. (Your syntax might vary. If it does, replace **monk_scripts/common** with your directory.)

Caution: *Do not specify absolute file-system paths when committing files to the e*Gate 4.5.3 Registry.*

To commit a single file to the e*Gate Registry

At the command line, type:

```
stcregutil -rh registryhost -rs schemaname -un egateusername
    -up password -fc repository_directory file_to_commit
```

For example, to commit the file **my_event.ssc** to the **monk_scripts/common** directory of the repository, type the following, making the appropriate substitutions for the host, schema, username, and password information:

```
stcregutil -rh registryhost -rs schemaname -un egateusername
    -up password -fc monk_scripts/common my_event.tsc
```

To commit multiple files to the e*Gate Registry

1. Copy all the files that you want to commit to the same directory. You can use a temporary directory if you prefer.
2. Use a text editor such as **Notepad** or **vi** to create a text file. Each line of the file should read as follows:

   ```
   filename,monk_scripts/common,FILETYPE_ASCIITEXT
   ```

   where ***filename*** is the name of the file you want to commit. Be sure that there are no spaces between commas and that the remainder of each line (***monk_scripts/common,FILETYPE_ASCIITEXT***) reads exactly as shown here.
3. Save the file with a **.ctl** extension (for example, **my_files.ctl**).
4. Type the following command to commit the files to the e*Gate Registry:

   ```
   stcregutil -rh registryhost -rs schemaname -un egateusername
       -up password -fc . -ctl ctl-file-name
   ```

3.4 Undefining the DGOS Environment Variable

The DGOS environment variable must be "undefined" in e*Gate 4.5.3 when using the Solaris 2.6/7 operating system. If the DGOS is set, the **stc_routablecollab.dll** will attempt to load the old DataGate **.dll** file, causing the system to return an error message saying that it cannot find the symbol.

In e*Gate 4.5.3, the environment variable is automatically supplied by the system, and no manual changes are required.

3.5 Making Configuration Changes to the Proxy e*Way

Using the **stcdgschema** command utility defines e*Ways and configures them to use a specific executable file (**stcewproxy.exe**) that performs the proxy functions. If desired, you can also make additional configuration changes (for example, to adjust the TCP/IP port range, or change debug flags).

To configure these parameters, you must use the e*Gate 4.5.3 Enterprise Manager's e*Way Editor GUI, modeled after the e*Gate 3.6 Superclient Editor. If you need more assistance using the e*Way Editor, see that GUI's online Help system or the *e*Gate Integrator User's Guide* found in the **/docs** directory of the e*Gate 4.5.3 installation CD-ROM disc 2.

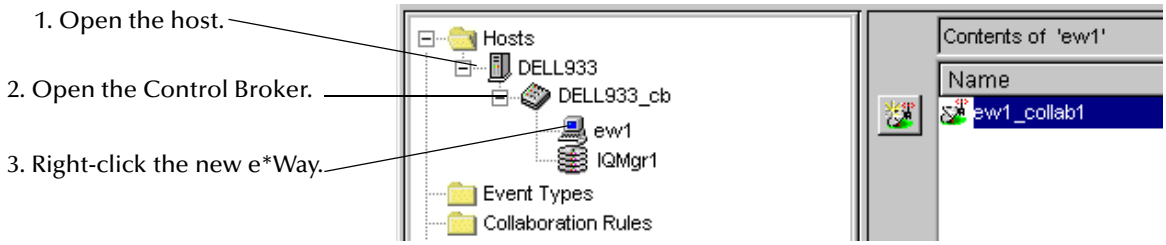
To configure a Proxy e*Way

1. If the Enterprise Manager is not already running, launch the Enterprise Manager. Log in to the Registry Host, and open the schema.

Note: *Changes are committed (saved) to the Registry as you make them.*

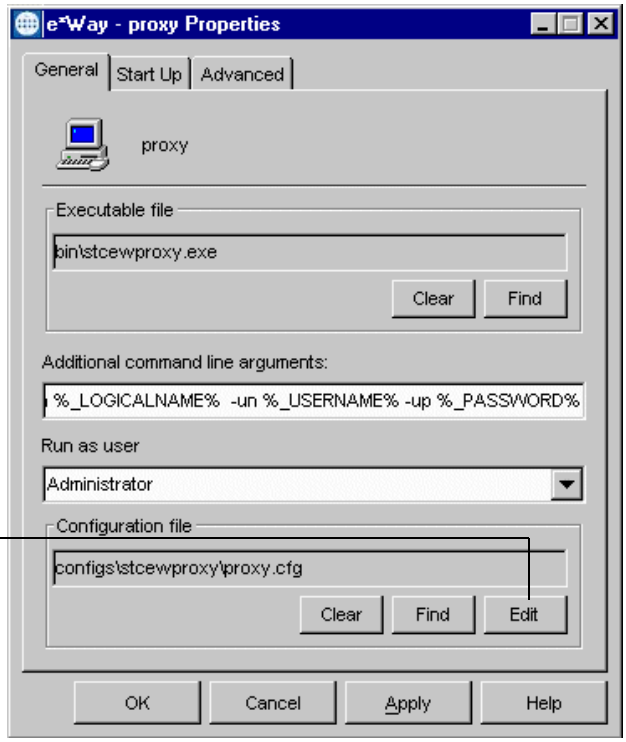
2. In the left pane of the Enterprise Manager, open the host, select the **Components** tab, and open the Control Broker under that host.
3. Right-click the new e*Way you want to modify. When the shortcut menu appears, click **Properties** (see **Figure 13 on page 38**).

Figure 13 Enterprise Manager with e*Way Displayed



4 The e*Way Properties dialog box appears. Under **Configuration File**, click **Edit** (see the following figure).

Figure 14 e*Way Properties Dialog Box



5 The e*Way Editor appears. Make any desired configuration changes.

6 After you have made all the necessary changes, click on the **File** menu and choose **Save**. You are prompted with a suggested file name (the same name as the e*Way that you are configuring).

7 Click **Save** to accept that default.

8 Exit the e*Way Editor.

You return to the e*Way Properties dialog box.

9 Click **OK** to apply your changes and exit the dialog box.

Running the e*Way: To correctly run a Proxy e*Way, you must set either of the following options:

- Set your PATH environment variable to include the location of the e*Gate 3.6 **bin** directory, where the Proxy e*Way executable files are stored.
- Enter the absolute path, along with the executable name, in the Proxy e*Way configuration file for the parameter **EXE name**.

3.6 Running dgw2ew.cmd

To replace the Proxy e*Ways with equivalent e*Gate 4.5.3 e*Ways, use the **dgw2ew.cmd** command utility to generate the new e*Way's configuration (**.cfg**) files. To activate this process, run the **dgw2ew.cmd** command utility.

Note: *Before running **dgw2ew.cmd**, you must have already created an e*Gate 4.5.3 schema using the **stcdgschema** command utility.*

The **dgw2ew.cmd** command uses the original e*Gate 3.6 e*Way configuration files to help generate equivalent e*Gate 4.5.3 e*Way files. You must run this command separately for each Proxy (3.6) e*Way you want to convert.

Important: *You can only run the **dgw2ew.cmd** utility under Windows.*

To run the dgw2ew.cmd command utility

1. Find the original **.sc** file in the e*Gate 3.6 directory structure for the e*Gate 3.6 e*Way to convert.
2. Copy the original **.sc** file into the appropriate e*Way configuration directory on the Windows system. In the table below, find your e*Gate 3.6 e*Way in the first column. Then, copy that e*Way's **.sc** file from its directory within the e*Gate 3.6 system to the directory in the e*Gate 4.5.3 system.

   In the following table, "eGate-" refers to the e*Gate 4.5.3 root directory (for example, **c:\eGate**).

Table 3 e*Way File Directory Locations

Converted e*Gate 3.6.x e*Way	e*Gate 4.5.3 directory to copy files into
Batch	eGate\client\configs\stcewgenericmonk\
COM/DCOM	eGate\client\configs\stcewgenericmonk\
CORBA Visibroker	eGate\client\configs\stcewgenericmonk\
Database access for ODBC	eGate\client\configs\stcewgenericmonk\
Database access for Oracle	eGate\client\configs\stcewgenericmonk\
Database access for Sybase	eGate\client\configs\stcewgenericmonk\
FTP	eGate\client\configs\stcewgenericmonk\
MQSeries	eGate\client\configs\stcewgenericmonk\
PeopleSoft Message Agent	eGate\client\configs\stcewgenericmonk\
SAP ALE	eGate\client\configs\stcewsapale\
SAP BAPI	eGate\client\configs\stcewgenericmonk\
SAP BDC	eGate\client\configs\stcewgenericmonk\
SAP EDI	eGate\client\configs\stcewsapedi\
Siebel EIM	eGate\client\configs\stcewgenericmonk\

Table 3 e*Way File Directory Locations (Continued)

Converted e*Gate 3.6.x e*Way	e*Gate 4.5.3 directory to copy files into
Siebel Event Driven	eGate\client\configs\stcewgenericmonk\
TCP/IP HL7	eGate\client\configs\stcdgwtcpiphl7

3 Change to the directory to which you copied the **.sc** file in the previous step.

4 Run the configuration file conversion utility. At the command prompt, type this command:

```
dgw2ew -s oldFileToConvert.sc -r RuleFileToUse.txt -d
    DefFileTemplateToUse.def
```

Where ***oldFileToConvert.sc*** is the file you copied in the first step above, and the other two arguments are found in the following table.

Table 4 e*Way Conversion Arguments

Converted e*Gate 3.6.x e*Way	Rule file to use	e*Gate 4.5.3 .def file template To use
Batch	batchFtpRule.txt	batch.def
COM/DCOM	com3.6To4.1Rule.txt	stcewms.def
CORBA Visibroker	corba3.6To4.1Rule.txt	stcewcorba.def
Database access for ODBC	dartRule.txt	dart.def
Database access for Oracle	dartRule.txt	dart.def
Database access for Sybase	dartRule.txt	dart.def
FTP	scFtp3.6To4.1Rule.txt	batch.def
MQSeries	mqseriesRule.txt	stcewmq.def
PeopleSoft Message Agent	pSoft3.6To4.1Rule.txt	psoft.def
SAP ALE	sapAleIn3.6To4.1Rule.txt *or* sapAleOut3.6To4.1Rule.txt	stcewsapalein.def *or* stcewsapaleout.def
SAP BAPI	sapBapi3.6To4.1Rule.txt	ewsapbapi.def
SAP BDC	sapBdc3.6To4.1Rule.txt	ewsapbdc.def
SAP EDI	sapEdiIn3.6To4.1Rule.txt *or* sapEdiOut3.6To4.1Rule.txt	stcewsapediin.def *or* stcewsapediout.def
Siebel EIM	siebelEim3.6To4.1Rule.txt	SiebelEim.def
Siebel Event Driven	siebelEvent3.6To4.1Rule.txt	stcewsiebeleventdriven.def
TCP/IP HL7	tcpiphl73.6To4.1Rule.txt	stcdgwtcpiphl7.def

Note: *The SAP ALE and SAP EDI e*Ways have both inbound and outbound rules files.*

The configuration utility creates **.cfg** and **.ctl** files, and a new **.sc** file, named after the **.sc** file you specified previously. The old **.sc** file is renamed and given an **.old**

extension. For example, if you are converting **my_file.sc**, the conversion utility creates **my_file.cfg**, **my_file.ctl**, a new **my_file.sc**, and renames the original file to **my_file.old**.

Important: *Remember these filenames; you need to specify them in later steps.*

5 Use the control file to commit the new files to the e*Gate 4.5.3 Registry. At the command prompt, type:

```
stcregutil -rh registryhost -rs schemaname -un username -up
    userpassword -fc . -ctl controlfilename.ctl
```

Where:

- ***registryhost*** is the name of the e*Gate 4.5.3 Registry Host.
- ***schemaname*** is the name of the e*Gate 4.5.3 schema that supports the migrated components. You can provide the name of your choice.
- ***controlfilename.ctl*** is the control file you generated in the previous step.

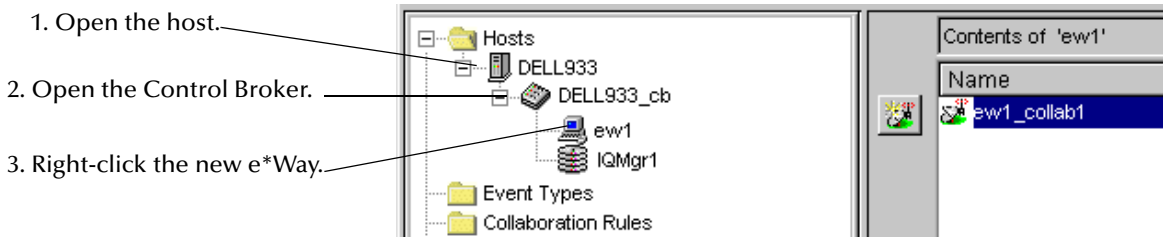
6 If the Enterprise Manager is not already running, open the Enterprise Manager, log in to the Registry Host, and open the schema.

Note: *Changes are committed (saved) to the Registry as you make them.*

7 In the left pane of the Enterprise Manager, open the host, and then open the Control Broker under that host.

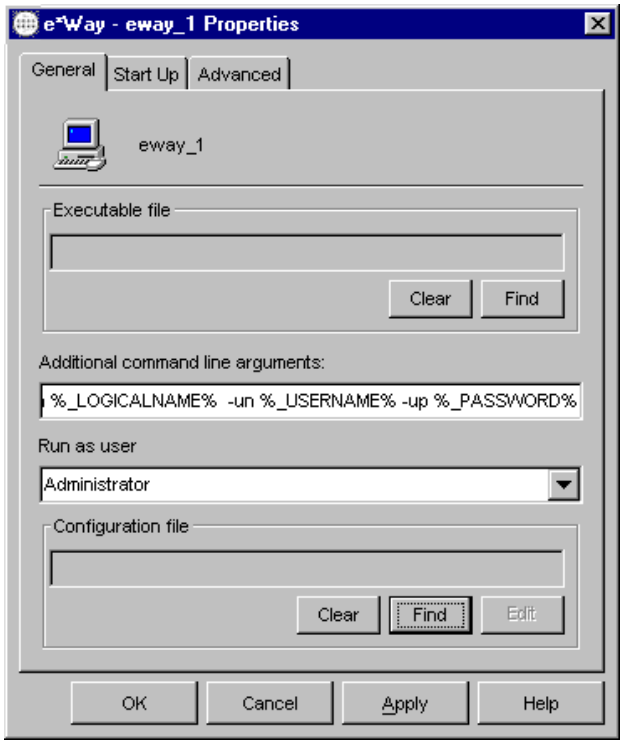
8 Right-click the new e*Way you want to modify. When the shortcut menu appears, click **Properties.** See the following figure.

Figure 15 Enterprise Manager with e*Way Displayed



The e*Way Properties dialog box appears. See the following figure.

Figure 16 Finding the e*Way Executable File



9 Replace the default executable file (**stcewproxy.exe**). Under **Executable File**, click **Clear** and then click **Find**.

A File selection dialog box appears.

10 Use the controls to select an executable file from the directory listing. Select the executable file according to the following table.

Table 5 e*Way Executable Files

Converted e*Gate 3.6.x e*Way	e*Gate 4.5.3 e*Way executable file
Batch	stcewgenericmonk.exe
COM/DCOM	stcewgenericmonk.exe
CORBA Visibroker	stcewgenericmonk.exe
Database access for ODBC	stcewgenericmonk.exe
Database access for Oracle	stcewgenericmonk.exe
Database access for Sybase	stcewgenericmonk.exe
FTP	stcewgenericmonk.exe
MQSeries	stcewgenericmonk.exe
PeopleSoft Message Agent	stcewgenericmonk.exe
SAP ALE	stcewsapale.exe

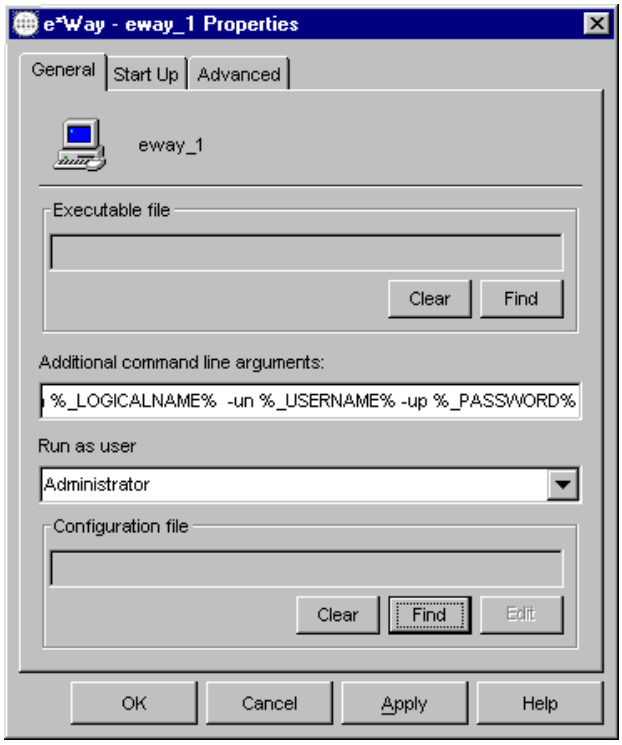
Table 5 e*Way Executable Files (Continued)

Converted e*Gate 3.6.x e*Way	e*Gate 4.5.3 e*Way executable file
SAP BAPI	stcewgenericmonk.exe
SAP BDC	stcewgenericmonk.exe
SAP EDI	stcewsapedi.exe
Siebel EIM	stcewgenericmonk.exe
Siebel Event Driven	stcewgenericmonk.exe
TCP/IP HL7	stcdgwtcpiphl7.exe

11 Once you have selected the file, click **Select** to close the File selection dialog box.

You return to the e*Way Properties dialog box (see the following figure).

Figure 17 Finding the e*Way Configuration File



12 Under **Configuration File**, click **Clear** and then click **Find**.

13 A File selection dialog box displays. Select the configuration (**.cfg**) file that you created in step 4 of this procedure.

14 Click **Select** to select the file and return to the e*Way Properties dialog box.

15 If you need to make any changes to the e*Way's configuration (see **"Testing and Troubleshooting" on page 47** for further information), click **Edit**. This action launches the e*Way Editor. Make any changes required, save the file, and exit the Editor. If you need assistance using the e*Way Editor, see the e*Way Editor's online Help system or the *e*Gate Integrator User's Guide*.

16 Click **OK** to close the e*Way Properties dialog box and save your configuration changes.

Note: *For information on how to change an encapsulated Route to a native Route in e*Gate 4.5.3, see* **Appendix C**.

3.6.1 Modifying DataGate Interfaces After Encapsulation

All changes to the DataGate encapsulated environment must be made in the old DataGate environment and moved to the new DataGate environment before the Proxy e*Ways can read the changes.

Note: *Some changes may also need modifications in the e*Gate environment.*

For data manipulation

1 For data manipulation, modify the original file (**.tsc**, **.ssc**, **.isc**, et cetera) by either editing the flat file or bringing up the ID, Structure, or X-late in the old GUI (if it is still available) and saving the changes.

or

If you are adding a new logic file (**.tsc**, **.isc**, et cetera), create the file as earlier practice allowed (GUI, if it is available, or flat file) and then add it to the **.rtb**, **.ttb**, **.atb**, **.mfl** files as necessary.

2 Move the file to the new DataGate environment (the environment that e*Gate reads). This environment is usually found on the new host for the e*Gate installation.

3 Stop and restart the e*Way (and subsequent DataGateWay) to read the new information.

For Route (Pub/Sub) changes

1 SeeBeyond strongly recommends that you create new e*Ways in e*Gate on a "go-forward" basis.

or

To modify an existing Route:

A Add the DataGateWay in the (old) DataGate environment.

B Add the new DataGateWay to the Route Table, including any **.isc** and **.tsc** pairs.

- Move the modified files to the new DataGate environment on the new host machine for e*Gate.
- Add additional Events and Event Types.
- Create the necessary pub/sub properties.
- Add additional Collaborations and Collaboration Rules to the existing Proxy e*Ways.

Stop and restart the Proxy e*Ways to ensure the changes are read.

Chapter 4

Testing and Troubleshooting

This chapter explains the recommended post-upgrade procedure for testing your new e*Gate 4.5.3 environment. Additionally, it describes information you need to know while upgrading from e*Gate 3.6.x e*Ways to their corresponding e*Gate 4.5.3 e*Ways.

4.1 Upgrading Configuration Parameters

You will need to refer to the procedures in this section only if you are substituting one of the following standard e*Gate 3.6/Proxy e*Way combinations to an e*Gate 4.5.3 equivalent:

- Batch or FTP

Note: *Both these e*Ways upgrade to the Batch e*Way.*

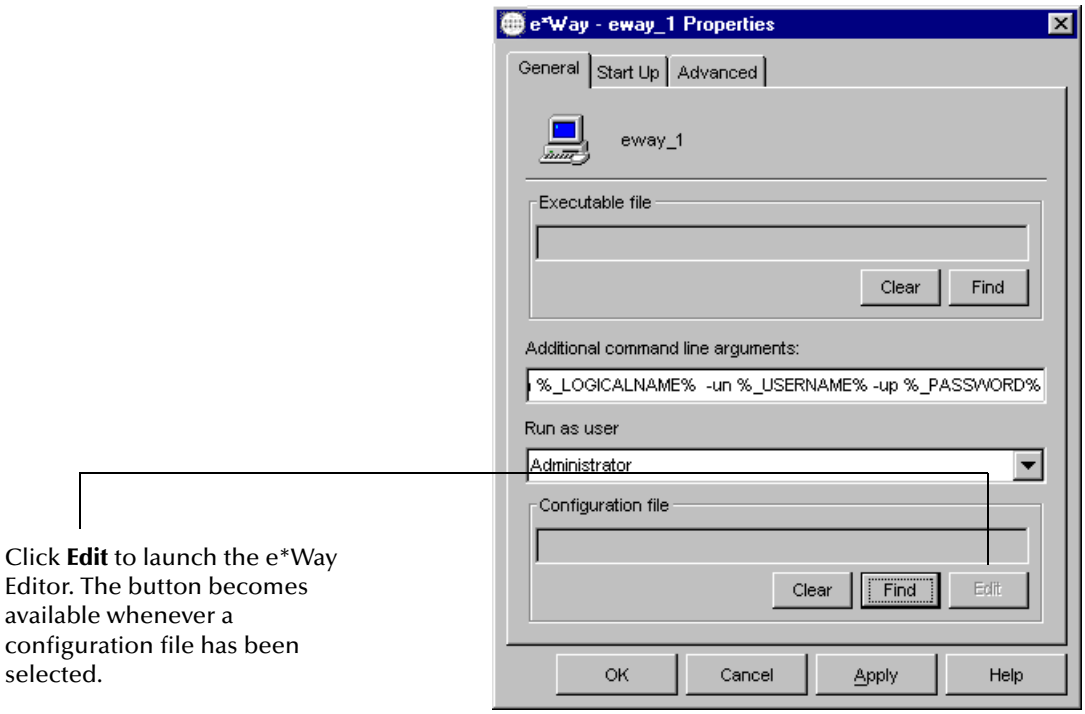
- Database access
- MQ Series

If you are not upgrading one of the standard e*Ways shown in the previous list, you can skip this section.

4.1.1 Using the e*Way Editor

The e*Gate 4.5.3 e*Way Editor is closely modeled after the e*Gate 3.6 SuperClient Editor. The e*Way Editor automatically opens when you edit a configuration file via an e*Way Properties dialog box. See **Figure 18 on page 48**.

Figure 18 e*Way Properties Dialog Box Showing the Edit Button



Note: *For more information about using the Enterprise Manager and the e*Way Editor, see their online Help systems or the* ***e*Gate Integrator User's Guide*** *in the* ***/Docs*** *directory on the e*Gate installation CD-ROM disc 2.*

4.1.2 Changes to all Monk-enabled e*Ways

The logging functions **dgw_log_*** are no longer supported. Instead, use the **display** function, and send the displayed string to the appropriate port for the desired logging level. Use these calls to define ports for different logging levels:

```
(set-port-callback (current-output-port) callback-stdout)
(set-port-callback (current-error-port) callback-stderr)
(set-port-callback (current-warning-port) callback-stdwarning)
(set-port-callback (current-debug-port) callback-stddebug)
```

4.1.3 Batch or FTP e*Ways

After conversion, both of these e*Ways require an encrypted password. If your configuration contains a password, you must re-enter it; the e*Way Editor performs the encryption for you. The password parameter is located in the **External Host Setup** section.

4.1.4 Database Access e*Ways

These e*Ways include those for Oracle, ODBC, and Sybase. For these e*Ways, you must complete the following procedures manually after the conversion:

1. In the e*Way Editor, go to the **Database Setup** section and modify the following items:
   - For the **Database Type** parameter, select the database type.
   - If the database requires a password, you must re-enter it in the **Encrypted Password** box. The e*Way now requires an encrypted password. When you re-enter it, the e*Way Editor performs the encryption for you.
2. In the e*Way Editor, go to the **Monk Configuration** section and modify the following items:
   - For the **Startup** function parameter, select **db-stdver-startup**. This function is a template that illustrates the required arguments and return codes for the e*Gate 4.5.3 version of this e*Way. You must modify that function (defined in the file **monk_library/dart/db-stdver-eway-funcs.monk**) to perform the same operations that your original startup function performed.

Note: *You cannot load the* ***.dll*** *file that the e*Gate 3.6 function used.*

   - For the **External Connection Establishment** function, select **db-stdver-conn-estab**. Like **db-stdver-startup**, this is a template function that you must modify, adding the functionality that your current **dbretry_com** function provides. The **db-stdver-conn-estab** function is also defined in the file **monk_library/dart/db-stdver-eway-funcs.monk**.
3. Any usage of **event-message** should be replaced with **event-send-to-egate**. Keep in mind that the new function returns a Boolean value.

4.1.5 MQ Series e*Ways

For these e*Ways, you must complete the following procedures manually after the conversion:

1. In the e*Way Editor, go to the **Monk Configuration** section and modify the following items:
   - For the **Startup** function parameter, select **MQ-stdver-startup**. This is a "template" function that illustrates the required arguments and return codes for the e*Gate 4.5.3 version of this e*Way. You must modify that function (defined in the file **monk_library/ewmsmq/msmq-startup.monk**) to perform the same operations that your original startup function performed.

Note: *You cannot load the* ***.dll*** *file that the e*Gate 3.6 function used.*

- If the e*Gate 3.6 message flow is set to **Always from DataGate** and the parameter **Xlate from DataGate** is blank, you must specify the e*Gate 4.5.3 parameter **Process Outgoing Message** function (in the Monk Configuration section) manually.
- If the e*Gate 3.6 message flow is set to **Always to DataGate** and the parameter **Xlate to DataGate** is blank, you must specify the e*Gate 4.5.3 parameter **Exchange Data With External** function (in the Monk Configuration section) manually.

2 Make the following changes within the **.isc** files:

- If you are using the MQ_SETTINGS_QMANAGER variable, replace it with MQ_SETTINGS_QUEUE_NAME.
- Replace **mgr-handle** with **MQ-mgr-handle**.
- Replace **queue-handle** with **MQ-queue-handle**.

4.2 Changes to e*Way-Specific Monk Functions

The tables in this section compare the e*Way-specific Monk functions that must be changed if you upgrade your e*Ways from e*Gate 3.6 to e*Gate 4.5.3. If you are using the RTC Service, you can disregard this section.

Note: *For more information about the Monk functions associated with an individual e*Way, see the user's guide for that particular e*Way.*

4.2.1 "%standard->julian" Monk Function is No Longer Valid

The "%" sign in Monk v4.x now has a more concise meaning. This requires converting all instances of the **%standard->julian** Monk function to **$standard->julian**.

4.2.2 Loop Syntax Change

The Loop syntax has changed; all instances of **[x]** must be replaced with **[<x>]**.

4.2.3 Database Access e*Way Functions

The following table shows a comparison of functions associated with these types of e*Ways, between e*Gate 3.6.1 and e*Gate 4.5.3.

Table 6 Database Access e*Way Functions — e*Gate 3.6.1 and 4.5.3

3.6.1	4.5.3	Changes
connection-handle?	connection-handle?	No changes.
db-alive	db-alive	No changes.
db-auto-bind-proc		No longer supported.
db-bind-proc	db-proc-bind	In e*Gate 4.5.3, the following parameters are not supported: **io-params, datatype-params, precision-params**, and **scale-params**. The e*Gate 3.6.1 and 4.5.3 versions of this function both bind the input/output parameters of the specified stored procedure. The e*Gate 3.6.1 version used an additional variable to collect information about each parameter.
db-call-proc	db-struct-call	In e*Gate 3.6.1, the **procedure-name** parameter accepted a string that specified the name of a procedure. In e*Gate 4.5.3, this parameter, now called **statement-handle**, accepts the handle of a stored procedure. In e*Gate 4.5.3, the **input-params** parameter has been replaced by the **procedure-path** parameter, which now accepts an absolute path to the procedure nodes in the Event Type Definition (ETD).
db-commit	db-commit	No changes.
db-get-error-str	db-get-error-str	No changes.
db-login	db-login	No changes.
db-logout	db-logout	No changes.
db-rollback	db-rollback	No changes.
db-sql-column-names	db-sql-column-names	No changes.
db-sql-column-types	db-sql-column-types	No changes.
db-sql-column-values	db-sql-column-values	Supports SQL_LONGVARxxx.
db-sql-execute	db-sql-execute	No changes.
db-sql-fetch	db-sql-fetch	No changes.

Table 6 Database Access e*Way Functions — e*Gate 3.6.1 and 4.5.3 (Continued)

3.6.1	4.5.3	Changes
db-sql-fetch-cancel	db-sql-fetch-cancel	No changes.
db-sql-select	db-sql-select	No changes.
db-sql-format	db-sql-format	No changes.
db-struct-fetch	db-struct-fetch	No changes.
db-struct-insert	db-struct-insert	No changes.
db-struct-select	db-struct-select	No changes.
db-struct-update	db-struct-update	No changes.
make-connection-handle	make-connection-handle	No changes.

Note: *These e*Ways include those for Oracle, ODBC, and Sybase.*

The following table shows a comparison of functions associated with these types of e*Ways, between e*Gate 3.6.2 and e*Gate 4.5.3.

Table 7 Database Access e*Way Functions — e*Gate 3.6.2 and 4.5.3

3.6.2	4.5.3	Change
connection-handle?	connection-handle	No change.
make-connection-handle	make-connection-handle	No change.
statement-handle?	statement-handle?	No change.
db-alive	db-alive	No change.
db-get-error-str	db-get-error-str	No change.
db-login	db-login	No change.
db-proc-bind	db-proc-bind	No change.
db-proc-column-count	db-proc-column-count	No change.
db-proc-column-name	db-proc-column-name	No change.
db-proc-column-type	db-proc-column-type	No change.
db-proc-execute	db-proc-execute	No change.
db-proc-fetch	db-proc-fetch	Supports SQL_LONGVARCHAR.
db-proc-fetch-cancel	db-proc-fetch-cancel	No change.
db-proc-param-assign	db-proc-param-assign	No change.
db-proc-param-count	db-proc-param-count	No change.
db-proc-param-io	db-proc-param-io	No change.
db-proc-param-name	db-proc-param-name	No change.
db-proc-param-type	db-proc-param-type	No change.

Table 7 Database Access e*Way Functions — e*Gate 3.6.2 and 4.5.3 (Continued)

3.6.2	4.5.3	Change
db-proc-return-exist	db-proc-return-exist	No change.
db-proc-return-type	db-proc-return-type	No change.
db-proc-return-value	db-proc-return-value	No change.
db-sql-column-names	db-sql-column-names	No change.
db-sql-column-types	db-sql-column-types	No change.
db-sql-column-values	db-sql-column-values	Supports SQL_LONGVARxxx.
db-sql-execute	db-sql-execute	No change.
db-sql-fetch	db-sql-fetch	No change.
db-sql-fetch-cancel	db-sql-fetch-cancel	No change.
db-sql-select	db-sql-select	No change.
db-sql-format	db-sql-format	No change.
db-struct-fetch	db-struct-fetch	No change.
db-struct-insert	db-struct-insert	No change.
db-struct-select	db-struct-select	No change.
db-struct-update	db-struct-update	No change.
db-struct-bulk-insert (Oracle & Sybase only)	db-struct-bulk-insert (Oracle & Sybase only)	No change.
db-struct-call	db-struct-call	No change.
db-proc-max-records (Oracle only)	db-proc-max-records (Oracle only)	No change.
db-std-timestamp-format	db-std-timestamp-format	No change.
db-commit	db-commit	No change.
db-logout	db-logout	No change.
db-rollback	db-rollback	No change.

Database Access e*Way Functions for e*Gate 4.5.3

The following list shows the database access e*Way functions for e*Gate 4.5.3:

db-proc-column-count	db-proc-column-name
db-proc-column-type	db-proc-execute
db-proc-fetch	db-proc-fetch-cancel
db-proc-max-records (Oracle only)	db-proc-param-assign
db-proc-param-count	db-proc-param-io
db-proc-param-name	db-proc-param-type

db-proc-return-exist	db-proc-return-type
db-proc-return-value	db-std-timestamp-format
db-stdver-conn-estab	db-stdver-conn-shutdown
db-stdver-data-exchg	db-stdver-init
db-stdver-init	db-stdver-neg-ack
db-stdver-pos-ack	db-stdver-proc-outgoing
db-stdver-proc-outgoing	db-stdver-shutdown
db-stdver-startup	db-stmt-bind
db-stmt-bind-binary	db-stmt-column-name
db-stmt-fetch	db-stmt-fetch-cancel
db-stmt-param-assign	db-stmt-param-bind
db-stmt-param-count	db-stmt-param-type
db-stmt-row-count	db-struct-bulk-insert (Oracle only)
event-send-to-egate	get-logical-name

4.2.4 Batch/FTP e*Way Functions

The following table shows a comparison of functions associated with this e*Way, between e*Gate 3.6 and e*Gate 4.5.3.

Table 8 Batch e*Way Functions — e*Gate 3.6 and 4.5.3

3.6	4.5.3	Changes
char-hex?	char-hex?	No change.
dgw_checkfor-commands		No longer supported.
dgw-ack-handler		No longer supported.
dgw-addto-periodic-scheduler		No longer supported.
dgw-addto-xlatetodg-scheduler		No longer supported.
dgw-at-schedule		No longer supported.
dgw-count-ack-received		No longer supported.
dgw-count-ack-sent		No longer supported.
dgw-count-message-received		No longer supported.
dgw-count-message-sent		No longer supported.
dgw-count-nak-received		No longer supported.
dgw-count-nak-sent		No longer supported.
dgw-debug-info		No longer supported.

Table 8 Batch e*Way Functions — e*Gate 3.6 and 4.5.3 (Continued)

3.6	4.5.3	Changes
dgw-disable-periodic-scheduler		No longer supported.
dgw-disable-xlatetodg-scheduler		No longer supported.
dgw-enable-periodic-scheduler		No longer supported.
dgw-enable-xlatetodg-scheduler		No longer supported.
dgw-exception		No longer supported.
dgw-exit		No longer supported.
dgw-extra-command		No longer supported.
dgwftp-append-path	ftp-append-path	"handle" added as an argument.
dgwftp-cleanup		No longer supported.
dgwftp-del-status		No longer supported.
dgwftp-expand	expand-string	No change.
dgwftp-fetch-files		No longer supported.
dgwftp-gen-done-file		DONE file generation not implemented in e*Gate 4.5.3
dgwftp-gen-done-filename		DONE file generation not implemented in e*Gate 4.5.3.
dgwftp-get-path		No longer supported.
dgwftp-get-seqno	get-seqno	In e*Gate 3.6, the file that contains the sequence number is a text file (**.seq**). In e*Gate 4.5.3, the file is a data file (**.dat**).
dgwftp-get-status		No longer supported.
dgwftp-incr-seqno	incr-seqno	No change.
dgwftp-load-configuration		No longer supported.
dgwftp-local-archive-file		No longer supported.
dgwftp-local-delete-file		No longer supported.
dgwftp-local-delete-file		No longer supported.
dgwftp-local-post-transfer	batch-local-post-transfer	In e*Gate 3.6, return values are Boolean **#t** or **#f**; in e*Gate 4.5.3, return values are undefined.
dgwftp-local-rename-file		No longer supported.
dgwftp-populate-queue		No longer supported.
dgwftp-put-path		No longer supported.

Table 8 Batch e*Way Functions — e*Gate 3.6 and 4.5.3 (Continued)

3.6	4.5.3	Changes
dgwftp-read-file		No longer supported.
dgwftp-recourse-action		No longer supported.
dgwftp-recover		No longer supported.
dgwftp-retry-wrapper		No longer supported.
dgwftp-rmt-archive-file		No longer supported.
dgwftp-rmt-delete-file		No longer supported.
dgwftp-rmt-list-files	ftp-rmt-list	"handle" added as an argument.
dgwftp-rmt-post-transfer	ftp-rmt-post-transfer	In e*Gate 3.6, this function determined the action to perform on a remote file after it had been processed by the e*Way. For inbound clients, this action was performed after the contents of the file had been processed and sent to e*Gate. For outbound clients, it was performed after the file had been transferred successfully. Beginning with e*Gate 4.5.1, this functionality changed to perform post-transfer operations on the specified file, depending on the setting of the **Local Command After Transfer** configuration parameter.
dgwftp-rmt-rename-file		No longer supported.
dgwftp-send-done-file		DONE file generation not implemented in e*Gate 4.5.3.
dgwftp-send-file	ftp-send	DONE file generation not implemented in e*Gate 4.5.3.
dgwftp-send-q-to-dg		No longer supported.
dgwftp-send-to-dg		No longer supported.
dgwftp-set-seqno	set-seqno	No change.
dgwftp-set-status		No longer supported.
dgwftp-string-is-proc?		No longer supported.
dgwftp-write-file		No longer supported.
dgw-get-configpath		No longer supported.
dgw-get-configstring		No longer supported.
dgw-get-configunsignedint		No longer supported.
dgw-get-seqnumber		No longer supported.

Table 8 Batch e*Way Functions — e*Gate 3.6 and 4.5.3 (Continued)

3.6	4.5.3	Changes
dgw-incr-seqnumber		No longer supported.
dgw-log-error		No longer supported.
dgw-log-info		No longer supported.
dgw-log-warning		No longer supported.
dgw-mark-periodic-appointment-fulfilled		No longer supported.
dgw-mark-xlatetodg-appointment-fulfilled		No longer supported.
dgw-monkret-externaldown		No longer supported.
dgw-monkret-msgrejected		No longer supported.
dgw-monkret-retry		No longer supported.
dgw-monkret-shutdown		No longer supported.
dgw-monkret-success		No longer supported.
dgw-monkret-success		No longer supported.
dgw-nak-handler		No longer supported.
dgw-notify-error		No longer supported.
dgw-replace-periodic-scheduler		No longer supported.
dgw-replace-xlatetodg-scheduler		No longer supported.
dgw-send-ack		No longer supported.
dgw-send-ifdown		No longer supported.
dgw-send-ifup		No longer supported.
dgw-send-message-to-dg		No longer supported.
dgw-send-nak		No longer supported.
dgw-set-configsection		No longer supported.
dgw-set-external-retry-seconds		No longer supported.
dgw-set-seqnumber		No longer supported.
dgw-shutdown		No longer supported.
dgw-skip-sending-message-to-dg		No longer supported.
dgw-startup	batch-init	Due to different architectures between e*Gate 3.6 and 4.5.3, different components are initialized, loaded, or set.

Table 8 Batch e*Way Functions — e*Gate 3.6 and 4.5.3 (Continued)

3.6	4.5.3	Changes
dgw-timestamp-lastreadfromexternal		No longer supported.
dgw-timestamp-lastwritetoexternal		No longer supported.
dgw-write-a-telnet-line		No longer supported.
dgw-xlate-from-dg		No longer supported.
dgw-xlate-to-dg		No longer supported.
expand-char	expand-char	No change.
expand-hex	expand-hex	No change.
expand-octal	expand-octal	No change.
expand-seqno	expand-seqno	No change.
expand-time	expand-time	No change.
fs-append-file	fs-append-file	No change.
fs-copy-file	fs-copy-file	No change.
fs-delete-file	fs-delete-file	No change.
fs-list-files	fs-list-files	No change.
fs-make-dir	fs-make-dir	No change.
fs-read-delim	fs-read-delim	No change.
fs-read-fixed	fs-read-fixed	No change.
fs-rename-file	fs-rename-file	No change.
ftp-append-file	ftp-append-file	"handle" added as an argument.
ftp-archive	ftp-archive	"handle" added as an argument.
ftp-archive-path	ftp-archive-path	"handle" added as an argument.
ftp-capture-data	ftp-capture-data	"handle" added as an argument.
ftp-change-dir	ftp-change-dir	"handle" added as an argument.
ftp-create-handle	ftp-create-handle	No default in e*Gate 4.5.3.
ftp-delete filename	ftp-delete	"handle" added as an argument.
ftp-delete-path	ftp-delete-path	"handle" added as an argument.
ftp-get-file	ftp-get-file	"handle" added as an argument.
ftp-get-last-response	ftp-get-last-response	"handle" added as an argument.
ftp-get-last-result-code	ftp-get-last-result-code	"handle" added as an argument.
ftp-get-path	ftp-get-path	"handle" added as an argument.
ftp-handle?	ftp-handle?	No change.
ftp-list-compare-size		No longer supported.

Table 8 Batch e*Way Functions — e*Gate 3.6 and 4.5.3 (Continued)

3.6	4.5.3	Changes
ftp-list-files	ftp-list-files	"handle" added as an argument.
ftp-list-raw	ftp-list-raw	"handle" added as an argument.
ftp-login	ftp-login	"handle" added as an argument.
ftp-make-dir	ftp-make-dir	"handle" added as an argument.
ftp-open-data-port	ftp-open-data-port	"handle" added as an argument.
ftp-open-host	ftp-open-host	"handle" added as an argument.
ftp-put-file	ftp-put-file	"handle" added as an argument.
ftp-put-path	ftp-put-path	"handle" added as an argument.
ftp-quit	ftp-ext-shutdown	No changes.
ftp-rename	ftp-rename	"handle" added as an argument.
ftp-rename-path	ftp-rename-path	"handle" added as an argument.
ftp-send-command	ftp-send-command	"handle" added as an argument.
ftp-send-reply-immediate	ftp-send-reply-immediate	"handle" added as an argument.
ftp-set-compare-time	ftp-set-compare-time	"handle" added as an argument.
ftp-set-debug		No longer supported.
ftp-set-handle		No longer supported.
ftp-set-mode	ftp-set-mode	"handle" added as an argument.
ftp-set-port		No longer supported.
ftp-set-timeout	ftp-set-timeout handle time	"handle" added as an argument.
read-file-complete		No longer supported.
read-file-delim		No longer supported.
read-file-fixed		No longer supported.
xgdbm-close		No longer supported.
xgdbm-delete		No longer supported.
xgdbm-fetch		No longer supported.
xgdbm-first-key		No longer supported.
xgdbm-handle?		No longer supported.
xgdbm-next-key		No longer supported.
xgdbm-open		No longer supported.
xgdbm-reorganize		No longer supported.
xgdbm-store		No longer supported.

Batch/FTP e*Way Functions for e*Gate 4.5.3

The following list shows the Batch/FTP e*Way functions for e*Gate 4.5.3:

batch-ext-verify
batch-nak
batch-proc-out
batch-rmt-post-transfer
batch-shutdown-notify
batch-startup
batch-validate-params
batch-write-file
connect-to-remote
disconnect-from-remote
fetch-files-from-remote
fetch-named-files
file-ext-connect
file-ext-shutdown
file-ext-verify
file-fetch
file-init
file-rmt-list
file-send
file-startup
file-validate-params
ftp-close
ftp-do-connect
ftp-ext-connect
ftp-ext-verify
ftp-fetch
ftp-heuristic-download
ftp-init
ftp-startup
ftp-validate-params
get-next-record
get-next-record-current-file
list-files-on-remote
open-next-working-file
persist-get-index
persist-get-list
persist-get-offset
persist-init
persist-read-number
persist-update-index
persist-update-list
persist-update-offset
persist-update-status
persist-write-pad
post-transfer-hook
pre-transfer-hook
send-files-to-remote
string-is-proc?
transfer-method?

4.2.5 MQSeries e*Way Functions

The following table shows a comparison of functions associated with this e*Way, between e*Gate 3.6.2 and e*Gate 4.5.3.

Table 9 MQSeries e*Way Functions—e*Gate 3.6.2 and 4.5.3

3.6.2	4.5.3	Changes
dgw-monkret-success		No longer supported.
dgw-monkret-failure		No longer supported.
dgw-monkret-retry		No longer supported.
dgw-monkret-msgrejected		No longer supported.
dgw-monkret-shutdown	shutdown-request	No change.
dgw-monkret-externaldown	send-external-down	No change.
dgw-set-configsection section_name_string		No longer supported.

Table 9 MQSeries e*Way Functions—e*Gate 3.6.2 and 4.5.3 (Continued)

3.6.2	4.5.3	Changes
dgw-get-configstring parameter_name_string default_value_string		No longer supported.
dgw-get-configpath parameter_name_string default_path_string		No longer supported.
dgw-get-configunsignedint parameter_name_string default_number min_number max_number		No longer supported.
dgw-send-message-to-dg message_string		No longer supported.
dgw-skip-sending-message-to-dg		No longer supported.
dgw-send-ifup		No longer supported.
dgw-send-ifdown		No longer supported.
dgw-send-ack		No longer supported.
dgw-send-nak		No longer supported.
dgw-checkfor-commands number_of_seconds		No longer supported.
dgw-debug-info info_string		No longer supported.
dgw-log-info info_string		No longer supported.
dgw-log-warning warning_string		No longer supported.
dgw-log-error error_string		No longer supported.
dgw-notify-error error_code error_string		No longer supported.
dgw-count-message-sent		No longer supported.
dgw-count-message-received		No longer supported.
dgw-count-ack-sent		No longer supported.
dgw-count-ack-received		No longer supported.
dgw-count-nak-sent		No longer supported.
dgw-count-nak-received		No longer supported.
dgw-get-seqnumber		No longer supported.
dgw-incr-seqnumber		No longer supported.
dgw-set-seqnumber number		No longer supported.

Table 9 MQSeries e*Way Functions—e*Gate 3.6.2 and 4.5.3 (Continued)

3.6.2	4.5.3	Changes
dgw-enable-xlatetodg-scheduler		No longer supported.
dgw-disable-xlatetodg-scheduler		No longer supported.
dgw-enable-periodic-scheduler		No longer supported.
dgw-disable-periodic-scheduler		No longer supported.
dgw-replace-xlatetodg-scheduler schedule_string-> status_code		No longer supported.
dgw-set-external-retry-seconds number of seconds		No longer supported.
dgw-write-a-telnet-line host_name_string port_number number _of_timeout_seconds telnet_line_string -> boolean		No longer supported .
MQCONN	MQCONN	No change.
MQOPEN	MQOPEN	No change.
MQBACK	MQBACK	No change.
MQBEGIN	MQBEGIN	No change.
MQCLOSE	MQCLOSE	No change.
MQCMIT	MQCMIT	No change.
MQCONNX	MQCONNX	No change.
MQDISC	MQDISC	No change.
MQGET	MQGET	No change.
MQPUT	MQPUT	No change.
MQPUT1	MQPUT1	No change.
MQ-configure-options	MQ-configure-options	No change.
MQ-create-config-value	MQ-create-config-value	No change.
MQ-init-type	MQ-init-type	The **MQobject** parameter was added to e*Gate in release 4.5.1. This parameter is of type *string*, and it accepts the name of the MQ data structure.

Table 9 MQSeries e*Way Functions—e*Gate 3.6.2 and 4.5.3 (Continued)

3.6.2	4.5.3	Changes
MQ-check-type	MQ-check-type	No change.
MQ-reset-type	MQ-reset-type	No change.
MQ-set-field	MQ-set-field	No change.
MQ-get-field	MQ-get-field	No change.

MQSeries e*Way Functions for e*Gate 4.5.3

The following list shows the MQSeries e*Way functions for e*Gate 4.5.3:

MQ-stdver-conn-estab
MQ-stdver-conn-shutdown
MQ-stdver-conn-ver
MQ-stdver-data-exchg
MQ-stdver-data-exchg
MQ-stdver-data-exchg-stub
MQ-stdver-init
MQ-stdver-neg-ack
MQ-stdver-pos-ack
MQ-stdver-proc-outgoing
MQ-stdver-proc-outgoing-stub
MQ-stdver-shutdown
MQ-stdver-startup
send-external-up
start-schedule
stop-schedule

Chapter 5

Frequently Asked Questions

This chapter lists some common questions that may be encountered during the e*Gate 3.6-to-4.5.3 upgrade operation, and the answers to those questions. It also provides a reference with tips, helpful hints, and best practices.

Important: *It is recommended that you read this chapter before beginning your upgrade.*

5.1 Introduction: Using These FAQs

The purpose of this chapter is to make you aware of some of the questions you should ask yourself before performing the upgrade operation. They are a combination of hints, tips, and ways to obtain optimum performance from your system. Hopefully, they will help you upgrade your system in the most efficient manner, and, at the same time, aid you in spotting problems to avoid.

5.2 Upgrade FAQs

This section answers commonly asked questions about general e*Gate upgrade operations.

5.2.1 Handling IQs and IQ Properties

1 **Is the *SYNC* option in e*Gate 4.5.3 the same as the *SYNC* option in DataGate?**

Yes. This flag determines whether the operating system (OS) writes every message to the disk, which the **DBSync = TRUE** in the JMS IQ Manager configurattion file or the OS caches messages in memory until it has an opportune time to write them to the disk (**SYNC = FALSE**). Message throughput performance is much faster when **SYNC = FALSE**, but there is a small window of time (the amount of time varies among different OSs) where there is message data in the OS cache memory. If the machine crashes, it is possible that data could be lost when you restart. The reliability of message delivery is improved with **SYNC = TRUE,** but the message throughput performance is slower.

Configure the **SYNC** flag in the IQ Manager Properties dialog box (in the Enterprise Manager). The default in **DBSync** in the JMS IQ Manager configuration file general settings is **TRUE**.

2 **What are Notification IQs?**

Notification IQs hold all resolved and observed Alert messages. The Control Broker actually keeps the unresolved messages in memory, so try to resolve all messages as soon as possible and set the **keep resolved Notification for** variable for Notification IQs down to a minimum for your site. Configure this variable in the Control Broker properties dialog box (Enterprise Manager).

3 **How do I set the *Event Type get interval* variable in the IQ properties dialog box?**

STC_Standard: For low-volume throughput for a particular IQ, set a higher wait time to reduce the number of polls to the IQ when it is empty. This setting can help conserve CPU cycles.

STC_JMS_IQ: For low-volume throughput for a particular IQ, set a lower wait time to reduce the number of polls to the IQ when it is empty. This setting can help conserve CPU cycles.

4 **Where do I set the expiration times for published Events in IQs? Do I leave these times unset for both active and journaled messages?**

Set the expiration times in the Collaboration Properties dialog box (Enterprise Manager). Set the journaled Event Type to expire at an appropriate time so that the IQ files do not get too large. By keeping the IQ files at a reasonable size, you can prevent using unnecessary disk space and CPU cycles.

5 **How many IQs do I configure per IQ manager?**

When using the **stcdgschema** utility, you configure as many IQs as there are outbound e*Way Intelligent Adapters. So a rule of thumb would be to divide the number of outbound e*Ways by three or four (your mileage may vary depending on throughput volume for a specific IQ). Then, round down any decimals and set the **-q** flag for the **stcdgschema** utility command to the resulting number. For example:

40 IQs/4 = 10 IQ Managers
Set **-q** to 10 using the **stcd gschema** command

50 IQs/4 = 12.5 IQ Managers
Set **-q** to 12 using the **stcdgschema** command

5.2.2 Table Mode Scripts

1 **Do the e*Gate Compatibility Components tools support Tables Mode scripts?**

Yes, with the RTC Service, Tables Mode scripts are supported. The Service can be used by e*Ways and/or BOBs.

5.2.3 Improving Performance and Throughput

1 **What is a JMS IQ? Can it help with my system's performance?**

The Java Message Service (JMS) IQ is a feature of e*Gate 4.5.3. Yes, it can help improve overall system performance. JMS IQs are designed to store data better in memory and on the disks. These IQs generally run faster than SeeBeyond Standard IQs. You can view and administer them using the JMS Administrator GUI.

For more information on the JMS IQ, see the *SeeBeyond JMS Intelligent Queue User's Guide*. For more information on subscriber pooling, see the *SeeBeyond eBusiness Integration Suite Deployment Guide*. For more information on the IQ Viewer, see the *e*Gate Integrator Intelligent Queue Services Reference Guide*.

5.2.4 Files, Scripts, and Monk Functions

1 **When upgrading from DataGate to e*Gate, are any files or scripts salvageable?**

Yes. The e*Gate 3.6 files and scripts with **.ssc**, **.tsc**, and **.dsc** extensions are compatible with e*Gate 4.5.3, which is more distributed and functionally robust than DataGate.

See SeeBeyond's support Web site at **http://www.seebeyond.com/services/support.html** for upgrade questions. The support site is protected, so you need a login name and password to access it. Once you have opened the page, click **Log in to Support** to log on.

If you do not have a name and password, click **Request Form**. After the **Service/Support Request Form** opens, fill out the pertinent information and click **Request**.

2 **When upgrading, are my custom Monk and Java functions reusable?**

Yes. SeeBeyond uses the Monk and Java programming languages to configure the various parts of the eBI Suite. All the custom Monk functions you created in earlier versions of e*Gate (including e*Gate 3.6) for Monk are forward-compatible. In e*Gate release 4.5.1 and later, you can program operations in Java as well as Monk.

3 **Why can't I get the Monk "set!" function to work?**

It does not work because the **set!** function is not capable of resolving the index of any repetition. In the example:

(set! current_var ~in%root.child[<i>.data)

the variable (**current_var**) is reset to a new value, but the **<i>** variable does not change.

This problem occurs in all the e*Gate upgrades (3.6.2, 4.1.1, 4.5.0 and 4.5.1), but has been corrected in release 4.5.2.

If the problem is encountered in any of the earlier upgrades, wrap the **get** function around the string. For example:

(set! current_var (get ~in%root.child[<i>.data))

5.2.5 Proxy e*Ways

1 **When upgrading, is it possible to save my custom Communication Clients or custom e*Ways?**

Yes. The Communication Client Proxy e*Way (**stcewproxy.exe**) enables a custom Communication Client e*Way or a standard e*Gate 3.6 e*Way to exchange data with e*Gate 4.5.3 via an Intelligent Queue (IQ). The Communication Client Proxy e*Way converts the e*Gate 3.6 communications protocol to the communications protocol used by e*Gate 4.5.3. This feature allows the Communication Client Proxy e*Way to either publish or subscribe to an IQ, enabling data to be transferred between the e*Gate 3.6 e*Way, the Communication Client Proxy e*Way, the IQ, and e*Gate 4.5.3.

5.2.6 Upgrading From e*Gate 4.X to e*Gate 4.5.1 or Later

1 **If I have e*Gate 4.X installed and I upgrade to e*Gate 4.5.1 or later, what will I have to do if I want to use the debug privilege?**

The debug privilege was added at e*Gate 4.5.1. If you use this privilege, you must take care when importing pre-4.5.1 schemas into the 4.5.1 or later releases, because they are imported without any debug privilege setting.

After importing such a schema, the Administrator must assign the debug privilege to all appropriate roles that should have debug access. The easiest way to do this is to import the **acl_roles.txt** file from a schema created in the current version of e*Gate (4.5.1 or later). This will work only if default roles/privileges have not been modified. Else, previous changes will be lost.

Note: *Be sure to follow the steps in the installation guides for the product or products you are installing. Pay special attention to provisos and caveats for customers upgrading from previous e*Gate releases.*

Appendix A

Upgrade Operation Checklist

This appendix provides a simple checklist you can use to guide and evaluate your e*Gate 3.6-to-e*Gate 4.5.3 upgrade operation.

A.1 Prerequisites

Before implementing the upgrade from the e*Gate 3.6 to an e*Gate 4.5.3 environment, complete the following general steps:

1. Upgrade the hardware as required by e*Gate 4.5.3 and as advised in the *e*Gate Integrator Installation Guide* and *SeeBeyond eBusiness Integration Suite Deployment Guide*.
2. Upgrade the necessary third-party software and operating systems as needed and as explained in the *e*Gate Integrator Installation Guide*.
3. Install e*Gate 4.5.3, including all applicable e*Way add-ons. See the *e*Gate Integrator Installation Guide* and one or more appropriate e*Way user's guides for more information.
4. If you are using any of the UNIX platforms, source the **egateclient.sh** or **egateclient.csh** file (as appropriate for your login shell) in the **client** subdirectory of the e*Gate directory tree.
5. Test your e*Gate 4.5.3 environment by creating the scenario documented in *Creating an End-to-end Scenario with e*Gate Integrator*.
6. Test your e*Gate 3.6 environment and verify that your existing e*Ways are fully operational.
7. Identify the e*Gate 3.6 e*Ways as either Proxy or e*Way 4.5.3 Configuration File Conversion candidates. See **"Supported e*Gate 3.6 e*Ways for Configuration File Conversion" on page 24** for a complete list. Specifically check translation scripts for any nonstandard monk functions that may not work in e*Gate 4.5.3.
8. If you have standard 3.6 e*Ways and are going to use the e*Way 4.5.3 Configuration File Conversion utility, identify all Monk scripts used in your e*Gate 3.6 e*Way. All scripts must be compatible with Monk version 4.5.3 standards (see **Chapter 4** for more information). Edit the scripts as needed.
9. Set up a test environment to run the upgrade.

A.2 Schema Construction

This section provides a list of the general steps required to build the new e*Gate 4.5.3 schema as follows:

1. Run the **stcdgschema** command for each e*Gate 3.6 environment being upgraded. See **Chapter 3** for details.
2. Check Collaborations for correct subscriptions and publications.
3. Confirm that the schemas and Proxy e*Ways were created correctly. Test to be sure that each Proxy e*Way and its related e*Gate 3.6 e*Way starts without any errors.
4. Test all Proxy e*Ways with data. In this test, include Proxy e*Ways that were created for e*Gate 3.6 e*Ways, which will be converted to e*Gate 4.5.3 e*Ways.
5. Convert any e*Gate 3.6 e*Way configurations to e*Gate 4.5.3 e*Ways using the **dgw2ew.cmd** procedure. See **"Running dgw2ew.cmd" on page 40** for details.
6. Test the new e*Gate 4.5.3 e*Ways with data.
7. Move the test environment to production then repeat the testing steps in these procedures before beginning production. See the *SeeBeyond eBusiness Integration Suite Deployment Guide* for guidelines on how to convert a test schema to a production schema.

Appendix B

Upgrading Monk e*Gate 3.6 to e*Gate 4.5.3

Monk e*Gate 4.5.3 contains a large number of functions not contained in Monk 3.6. See the release notes for both Monk release 4.0.1 and Monk e*Gate 4.5.3 for more information. For additional details, see the *Monk Developer's Reference* for e*Gate 4.5.3.

This appendix provides a summary of these changes.

B.1 Pathing Strictness Increased (~ and %)

The usage of ~ and % has been modified.

~MSH.ENV indicates the name of the event definition

%MSH.ENV indicates a partial path

Monk functions using % to perform modulo operations are no longer acceptable.

B.2 New Tokens

Tokens are now recognized uniquely:

. , ,@ ...

This has been done to enable the implementation of a true quasi quote.

B.3 Dot "." as Initial Character

Dot "." is no longer accepted as the initial character of an identifier.

B.4 64 Bit File Type

The 64 bit file type is supported. Also unsigned integers are supported; however, not all platforms support large numbers.

B.5 Non-printable Values

All non-printable values are in hex instead of octal.

B.6 Dynamic Parsing of Data

When adding data to an existing child node, data present in its parent node is marked invalid.

When data is written to a child node that does not exist, the data is parsed from the parent node into the child nodes. Data is added to the child node, and data in the parent node is marked invalid.

When data is added to a parent node, but the parent node does not contain valid data, the following actions happen:

- Data is reconstituted from the child nodes.
- The child sub-tree is deleted.
- Data is added to the parent node.

When data is added to a parent node, and the parent node contains valid data, the following actions happen:

- The child sub-tree is deleted.
- Data is added to the parent node.

When data is written to the set node, it automatically forces a parse/reparse of the child nodes.

B.7 Dynamic Extension of Event Definitions

If the Event has delimiters beyond what has been defined in the event definition, the Event definition extends to encompass the additional delimiters. To capture this behavior accurately, the following new functions have been created:

- **count-map-children**
- **count-data-children**

B.8 Modification of the Monk Parser (Dotted Pair)

The Monk Parser recognizes **(a . b)**, and creates a true dotted pair.

B.9 Delimiters

- Multiple begin/end delimiter pairs are supported.
- A seperator is no longer valid as a delimiter modifier.

B.10 Behavior of Optional Nodes

Previously, input tags were inserted in optional nodes with no data to prevent these nodes from being output as empty nodes for delimited Event structures. Empty strings would be generated to keep the input and output stream consistent.

The behavior of optional untagged nodes has been modified to better handle the output of data. Promotion attributes are assigned to map nodes based on parent-sibling and sibling-sibling relationships.

A *strongly unique node* (required tagged) promotes all preceding siblings from Non-Unique (NU) to Required Non-Unique (RNU).

A *strongly unique child node* promotes its parent from Non-Unique to Weakly-Unique (WU) type of node.

A *required non-unique node* (required untagged) promotes all preceding Non-Unique siblings to Required Non-Unique type nodes.

A *strongly unique child* promotes its parent from Required Non-Unique node to Strongly Unique node type.

The following table illustrates the node types that support possible child-to-parent Promotions. Sibling nodes trailing their predecessors can also trigger promotions in node type at run time.

Table 10 Node Types Supporting Promotions

Node Type	Parent<-child promotion	Predecessor siblings<-trailing sibling promotion
RNU (required untagged node)	SU <-SU None <-RNU None <- WU None <- NU	None
NU (optional untagged node)	WU <- SU None <- RNU None <- WU None <- NU	RNU <- SU RNU <- RNU None <- WU None <- NU

The acronyms in the previous table have the following definitions:

RNU (required, non-unique)
required, untagged

SU (strongly unique)
required, tagged

WU (weakly unique)
optional, tagged

NU (non-unique)
optional, untagged

B.10.1 Nodes Generated As Output Nodes

Applying the promotional behavior, the final attributes of the nodes are determined (see the following table).

Table 11 Nodes Generated as Output

Node type	Generates an output node
SU	Yes
NU	Yes
WU	No
RNU	Yes

B.10.2 Order of Promotional Behavior

The first phase of the promotional behavior occurs when the Event Type Definition is created by the user (the function **$make-event-map** is called), but no data has been mapped.

Node promotion in this example is in the following order:

1. Sibling to sibling promotion, to determine which nodes are output.
2. Child to parent promotion, which supersedes the sibling to sibling promotion.

This process determines a final Event map tree structure.

At runtime, when data is passed through this map tree, another type of promotion takes place.

If a node is Non Unique (optional and untagged) and has data in any of its trailing siblings (that is: at the same node level), an output node is generated.

If the above condition is not fulfilled, the output nodes are generated only as the result of the sibling-sibling and parent-child promotions.

B.11 New Interface Functions

The two new functions, **load-interface** and **invoke**, enable greater interface capabilities to functions written in languages other than Monk. See the *Monk Developer's Reference* for further details.

B.12 Immutability

Vectors, lists, and strings cannot be modified if they are constants. For example, the following expression fails:

```
(define foo "fixed")
(string-set! foo 2 #)
```

B.13 Fixed Node Data Definition

For a fixed node, the data is exactly defined by the input tag. The input tag denotes the limits of the data (content and length). The result of this match is the data. An input tag in the parent node defines the data limits for its child nodes to map.

BdB and **Pp** were created to allow backward compatibility.

B.14 HL7 Functions

The following functions have either been replaced or modified:

- **name->hl7name**
- **date->hl7date**
- **time->hl7time**
- **event-send**
- **message-send**
- **RPC Functionality**

B.15 Exception Support

All prior references to errors in Monk have now been replaced by exceptions. The following Monk functions/constructs help implement exceptions:

- **always**
- **define-exception**
- **exception-symbol**
- **exception-category**
- **exception-string**
- **exception-string-all**
- **try**
- **catch**
- **throw**

Exception messages

Exception messages can now be passed through the C API.

B.16 The regex Functions

In Monk 4.0, **regex** functions as documented. As a result, any work-arounds for **regex** may not work as expected.

B.17 New Modifiers

- **BdB** allows backward compatibility. Begin delimiter bind. Designates that if you have a begin delimiter, you must have a matching end delimiter from the same pair.
- **EscD** is an escape delimiter that gives the character following the character that is escaping a special meaning.
- **ExF** specifies that you cannot expand the data map. If the data exceeds the map, it will fail and not map any of the data.
- **Pp** allows backward compatibility. Parent precedence. The parent delimiter will take precedence over the child-node delimiters.

Note: *For additional information about modifiers see the* ***Monk Developer's Reference****.*

B.18 Modified Monk Functions

- **append** now accepts multiple arguments.
- **read** now returns End Of File object on reading End Of File.
- **readline** now returns End Of File object on reading End Of File.
- **change-pattern** has been fixed to operate as stated in the Monk documentation.
- **sprintf** was removed.
- Fixes for **set** manipulations, making them consistent with the Monk documentation.
- Return values for **set!**, **set-car**!, and **set-cdr!** have been modified to no longer return an unspecified value. The value of the variable bound to by the functions is now returned.
- **lambda** form modifications/additions.
- **define** form modifications/additions.
- **quasiquote** functionality implemented.
- **assoc** now verifies all arguments.
- **string-port->string** has an additional parameter for better memory usage.
- **number->string** returns **False** if it cannot convert the number to the chosen base type.
- **string->number** returns **False** if it cannot convert the number to the chosen base type.

Note: *For more information, see the* ***Monk Developer's Reference****.*

B.19 Functions No Longer Supported

The file **no_longer_supported.monk** contains all the functions that are no longer supported as monk aliases until they are phased out completely. **Table 12 on page 77** gives a list of nonsupported functions.

Table 12 Non-supported Monk Functions in e*Gate 4.5.3

e*Gate 3.6 functions	Corresponding e*Gate 4.5.3 functions
count-children	Replaced by **count-map-children** and **count-data-children**. **Note:** All references to **count-children** point to **count-map-children**. The **count-children** function is identical to the **count-data-children** function. Code using the **count-children** function still runs correctly, but to avoid future problems, change at the earliest opportunity to refer to **count-data-children**.
defined-as-repeating	Replaced by **path-defined-as-repeating?**.
get-node-depth **Note**: Has been Fixed to return the correct depth.	Replaced by **path-nodedepth**.
IO-to-lower	Replaced by **string-downcase**.
node-name	Replaced by **path-nodename**. **Note**: Verifies the root node as being valid if a number is used.
seek	Replaced by **seek-cur**.
strip-trailing-whitepace	Replaced by **string-trim**.
trim-char-left	Replaced by **string-left-trim**.
trim-char-right	Replaced by **string-right-trim**.
close-append-port	Replaced by **close-port**.
close-input-port	Replaced by **close-port**.
close-input-string	Replaced by **close-port**.
close-output-port	Replaced by **close-port**.
close-output-string	Replaced by **close-port**.
close-random-access-port	Replaced by **close-port**.
count	Replaced by **count-rep**. **Note**: For backward compatibility purposes, all previous usage of count is still evaluated by **count-rep**.
message-send	Replaced by **event-send-to-egate**.
insert	Replaced by **copy**.
insert-hard	Replaced by **copy**. **Note:** Replaces the data in position with the data that is given. The data does not have to be the same length.
read-line	Still **read-line**, but now it reads the **eof** object, whereas before it could not.

Table 12 Non-supported Monk Functions in e*Gate 4.5.3 (Continued)

e*Gate 3.6 functions	Corresponding e*Gate 4.5.3 functions
message-clear	Replaced by **$event-clear**.
message-parse	Replaced by **$event-parse**.
message->string	Replaced by **$event->string**. **Note**: Adds an additional parameter to allow for better memory usage; it remembers the last size that was used.
make-message-structure	Replaced by **$make-event-map**.
message-convert	Replaced by **$resolve-event-definition**.
sub-string **Note:** Can still call **sub-string** for backward compatibility.	Replaced by **sub-sequence**
to-upper	Replaced by **string-upcase**.
trim-space	Replaced by **string-trim**.
trim-space-left	Replaced by **string-left-trim**.
trim-space-right	Replaced by **string-right-trim**.

B.20 Real Number Precision

Real number precision is no longer limited to 13 digits to the right of a decimal point; it now can have up to 21 digits.

Note: *Not all platforms support large numbers.*

B.21 UTF8 and UTF16

Both UTF8 and UTF16 are supported.

Additional support for UTF8 conversion is provided through the UTF8 Conversion utility—**utf8convert.exe**. Use the UTF8 conversion utility to convert Collaboration Rules Scripts (**.tsc**), Event Type Definitions (**.ssc**), and XML files into UTF8 format.

The UTF8 Conversion utility is located in:

- **/eGate/client/bin/**

Note: *For more information, see the* ***Monk Developer's Reference****.*

Appendix C

Moving Encapsulated Routes to Native

This appendix explains how to move an encapsulated Route to a native Route in e*Gate 4.5.3, after upgrading.

C.1 Moving to Native Routes: Introduction

Once you have upgraded from e*Gate 3.6 to e*Gate 4.5.3 you are now on a supported platform and version. The new integration of new and existing interfaces can be added to your schema.

You may have had a need to reimplement some of your existing encapsulated Routes and interfaces but did not have the time or resources to accomplish this before your upgrade. When you do have time and resources available, you can reimplement any of your encapsulated Routes and interfaces as defined by this appendix.

C.2 Moving to Native Routes: Procedure

Before beginning this procedure you must rewrite the Tables code logic with the Monk programming language (Tables mode only).

In Monk only, you can reuse:

- Event Type Definition (ETD) logic
- ID logic
- Xlate logic
- Custom Monk functions

To move an encapsulated Route to a native Route

1. Identify the script language as follows:
   - Tables mode: You must rewrite Tables mode logic with Monk.
   - Monk mode: You can reuse **.ssc**, **.tsc** and **.isc** files.

Note: *See* **Appendix B** *for details on new Monk function names. All* ***.ssc****,* ***.tsc****, and* ***.isc*** *files must have new 4.x Monk equivalents before they can work. For example,* ***message-parse*** *becomes* ***$event-parse*** *and* ***make-message-structure*** *becomes* ***$make-event-map****.*

2 Disable the encapsulated Route as follows:

- Identify the Route you want to disable.
- Find the publications and subscriptions.
- Remove e*Gate 3.6/Proxy e*Way combinations.
- Replace each combination with the equivalent e*Gate 4.5.3 e*Way (using the same logical name).

Note: *Do not delete these e*Ways. You do need to rename the e*Ways' executable files. Also, modify the configurations and the Collaboration Rules in the Collaborations. The e*Ways' logical names stay the same.*

- Modify the executable and configuration files using the Enterprise Manager's e*Way Editor.
- If you are in the Tables mode:
  - Rewrite the Tables logic in Monk.
  - Point to the new **.ssc**, **.tsc**, and **.isc** files.
- If you are already in the Monk mode:
  - Update/modify any e*Gate 3.6 Monk to the e*Gate 4.5.3 Monk version.
  - You must consider ID logic to determine the appropriate places to do Xlates.

3 Add the native Route as follows:

- If you are in the Tables mode:
  - Use the new **.ssc**, **.tsc**, and **.isc** files.
- If you are already in the Monk mode:
  - Reuse the **.ssc** (ETD) files.
  - Reuse the **.isc** and **.xsc** (Collaboration Rules) files.

Appendix D

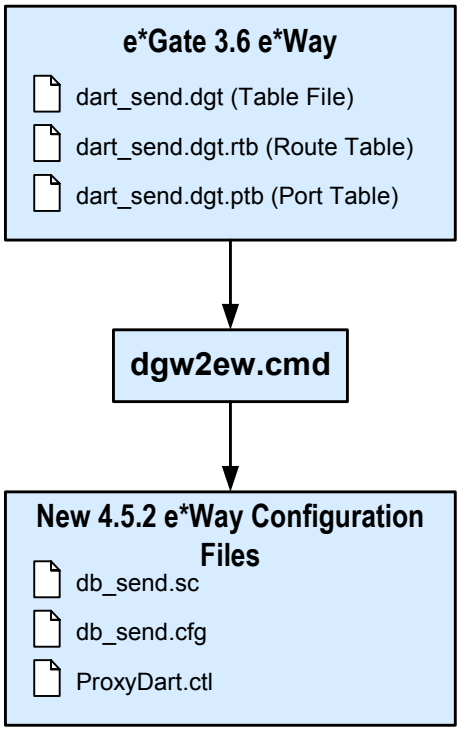
Sample Configuration Files

This appendix shows the contents of sample configuration files in e*Gate both before and after upgrading from e*Gate 3.6 to e*Gate 4.5.3.

D.1 Files: Schema Configuration Utility

This section contains samples of the configuration files shown under **"Schema Configuration Utility" on page 17**. Refer to this sample to gain an understanding of the contents of the original files and that of the converted e*Way e*Gate 4.5.3 files. The following figure provides an illustration.

Figure 19 New e*Way Configuration Files



The rest of this section shows examples of the contents of the files shown in the previous figure.

D.1.1 dart_send.dgt

The following example shows an e*Gate 3.6 Table File:

```
//dgEdit 3.0
#include "dart_send.dgt_dgt/dart_send.dgt.atb"
#include "dart_send.dgt_dgt/dart_send.dgt.ptb"
#include "dart_send.dgt_dgt/dart_send.dgt.rtb"
#include "dart_send.dgt_dgt/dart_send.dgt.mtb"
#include "dart_send.dgt_dgt/dart_send.dgt.ttb"
__MONKFILE="dart_send.dgt_dgt/dart_send.dgt.mfl";
```

D.1.2 dart_send.dgt.rtb

The following example shows an e*Gate 3.6 Route Table file:

```
//dgEdit
route_table_t route_table {
//
//
// Logical     Message_ID    Destination    Translation
//
   "db_send",  "DG_ID_ALL",  "eater",       "",
   "feeder",   "DG_ID_ALL",  "db_send",     "",
   "",         "",           "",            ""
}
```

D.1.3 dart_send.dgt.ptb

The following example shows an e*Gate 3.6 Port Table file:

```
//dgEdit
port_table_t port_table {
//
//
// Logical     Physical
Direction    Executable     RAM Queue
//
  "db_send", "C:\\SeeBeyond\\datagate\\configs\\ScDb\\db_send.cfg",
PORT_IO,     "ScDb.exe",         -1,
  "eater",   "C:/SeeBeyond/datagate/configs/eater/dart_eaterM.cfg",
PORT_O,      "eater.exe",        -1,
   "feeder",   "C:/SeeBeyond/datagate/configs/feeder/
dart_feeder2M.cfg",  PORT_I,      "feeder.exe",       -1,
   "",         "",                                                    -
1,         "",               -1
}
```

D.1.4 db_send.sc

The following example shows the new **.sc** file for the Proxy Communication Client e*Way:

```
; ------------------------------------------------------------------
-------
;               General Info
; ------------------------------------------------------------------
-------
(general-info
  (version "E*Gate")
```

```
  (revision "$Revision: 1.3.4.2.2.4 $")
  (user "$Author: btu $")
  (modified "$Date: 2000/05/02 21:43:06 $")
  (creation "initial")
  (description "Old DGW Proxy Module:

This E*Gate Module allows old DGWs to communicate to it
and assume that is it connecting to DataGate 3.X server.

")

  (user-comment "")
  (generated-cfg-path "")
  (delim1 '\n')
  (delim2 '|')
  (delim3 '=')
  (delim4 ',')
  (cfg-icon "")
)
; ------------------------------------------------------------------------
-------
;                Super Client Type
; ------------------------------------------------------------------------
-------
(super-client-type
  (network-protocol     "<ANY>")
  (os-platform          "<ANY>")
  (protocol-api-version "<ANY>")
  (app-protocol         "<ANY>")
  (direction            "<ANY>")
)
; ------------------------------------------------------------------------
-------
;   Section:"General Settings"
; ------------------------------------------------------------------------
-------
(section
  (name "General Settings")
    (string-set
        (name "Exe Name")
        (value "")
        (config-default "")
        (set
            (value ())
            (config-default ())
        )
        (description "Exe Name:
This is the name of the datagate e*Way executable such as
ScFtp.exe, eater.exe, etc...

")
        (user-comment
            (value "")
            (config-default "")
        )
    )

    (string-set
        (name "CFG Name")
        (value "")
        (config-default "")
        (set
            (value ())
            (config-default ())
```

```
        )
        (description "CFG Name:
This is the name of the configuration file for datagate e*Way
with the full path.
        ex: c:\data\stc\tables\ftp.cfg
")
        (user-comment
            (value "")
            (config-default "")
        )
    )

    (string-set
        (name "Debug Flags")
        (value "")
        (config-default "")
        (set
            (value ())
            (config-default ())
        )
        (description "debug flags:
This is the debug flags that is used to pass in when the
proxy invoke datagate e*Way

        ex: ffffffff

")
        (user-comment
            (value "")
            (config-default "")
        )
    )

    (int-set
        (name "Lower IP Port")
        (value none)
        (config-default none)
        (set
            (value ())
            (config-default ())
        )
        (range
            (value (const 2000 const 65536))
            (config-default (2000 65536))
        )
        (description "Lower IP Port:

This is a starting IP Port for the DGW to use to connect to the Proxy
for both datagate server and NCB port.
The Proxy will start as this port and try to bind the first in the
range of ports between the lower and upper IP Port.

See Also: Upper IP Port

")
        (user-comment
            (value "")
            (config-default "")
        )
    )

    (int-set
        (name "Upper IP Port")
        (value none)
```

```
            (config-default none)
            (set
                (value ())
                (config-default ())
            )
            (range
                (value (const 2000 const 65536))
                (config-default (2000 65536))
            )
            (description "Upper IP Port:

        This is a ending IP Port for the DGW to use to connect to the Proxy.
        The Proxy will start at the lower port and try to bind the first in
        the range of ports between the lower this upper IP Port.

        See Also: Lower IP Port

        ")
            (user-comment
                (value "")
                (config-default "")
            )
        )

        (string-set
            (name "Daemonize")
            (value const "NO")
            (config-default "NO")
            (set
                (value const ("YES" "NO"))
                (config-default ("YES" "NO"))
            )
            (description "Daemonize:

         If this parameter is set to YES then the old e*Way
         will daemonized.
        ")
            (user-comment
                 (value "")
                 (config-default "")
            )
        )

        (string-set
            (name "Stall Daemonize")
            (value const "NO")
            (config-default "NO")
            (set
                (value const ("YES" "NO"))
                (config-default ("YES" "NO"))
            )
            (description "Daemonize:

         If this parameter is set to YES then the old e*Way
         will stalled for 20 seconds.
        ")
            (user-comment
                (value "")
                (config-default "")
            )
        )

        (string-set
            (name "Auto Start Executable")
```

```
            (value const "YES")
            (config-default "YES")
            (set
                (value const ("YES" "NO"))
                (config-default ("YES" "NO"))
            )
            (description "Auto Start Executable:

     If this parameter is set to YES then the proxy will start
    the configured executable automatically.  Set this parameter to NO to
    manually
    start the executable locally or remotely.  This is useful when
    the executable is compiled in an older OS version or run in a
    different OS from the proxy.

    Note: When starting the executable manually, the user needs to find
    out
          the DG port number that the proxy binds to in the proxy log
    file.
          Look for \"bound DG IP port\" and use this value to start the
    executable.
    ")
            (user-comment
                (value "")
                (config-default "")
            )
        )

        (description "General Settings:

        EXE Name
        CFG Name
        Debug Flags
        Lower IP Port
        Upper IP Port
        Daemonize
        Stall Daemonize
            Auto Start Executable
    ")

      (user-comment
        (value "")
        (config-default "")
      )
    )
```

Note: *See the* ***Communication Client Proxy e*Way Intelligent Adapter User's Guide*** *for information on how to start the Proxy e*Way manually.*

D.1.5 db_send.cfg

The following example shows the new **.cfg** file for the Proxy e*Way:

```
#
# ------------------------------------------------------------------
-------
#               Delimiters To Use
# ------------------------------------------------------------------
-------
#
File/CFG/Version:0.0/Delim1:\o012/Delim2:\o174/Delim3:\o075/
Delim4:\o054
```

```
#
#
# ------------------------------------------------------------------------
-------
#                General Info
# ------------------------------------------------------------------------
-------
#
#  version:E*Gate
#  revision:$Revision: 1.1.2.32 $
#  user      :$Author: lealon $
#  modified:$Date: 2000/05/17 23:43:23 $
#  creation:initial
#
#
# ------------------------------------------------------------------------
-------
#                 DataGateWay Type
# ------------------------------------------------------------------------
-------
#
#  network-protocol:<ANY>
#  os-platform:<ANY>
#  protocol-api-version:<ANY>
#  app-protocol:<ANY>
#  direction:<ANY>
#
#
# ------------------------------------------------------------------------
-------
#   Section:General Settings
# ------------------------------------------------------------------------
-------
#
General Settings|Exe Name|value=D:/SeeBeyond/datagate/Win32/bin/
feeder.exe|set=D:/SeeBeyond/datagate/Win32/bin/feeder.exe
General Settings|CFG Name|value=D:/SeeBeyond/datagate/tables/
DemoUpgrade.dgt_dgt/feeder1.cfg|set=D:/SeeBeyond/datagate/tables/
DemoUpgrade.dgt_dgt/feeder1.cfg
General Settings|Debug Flags|value=0|set=0
General Settings|Lower IP Port|value=5000|set=5000|range=2000,65536
General Settings|Upper IP Port|value=5500|set=5500|range=2000,65536
General Settings|Daemonize|value=NO|set=NO,YES
General Settings|Stall Daemonize|value=NO|set=NO,YES
General Settings|Auto Start Executable|value=YES|set=NO,YES
```

D.1.6 ProxyDart.ctl

The following example shows the control (.**ctl**) file created for the Proxy e*Way:

```
db_send.sc,configs/stcewproxy,FILETYPE_ASCIITEXT
db_send.cfg,configs/stcewproxy,FILETYPE_ASCIITEXT
eater.sc,configs/stcewproxy,FILETYPE_ASCIITEXT
eater.cfg,configs/stcewproxy,FILETYPE_ASCIITEXT
feeder.sc,configs/stcewproxy,FILETYPE_ASCIITEXT
feeder.cfg,configs/stcewproxy,FILETYPE_ASCIITEXT
```

D.2 Files: e*Way Configuration File Conversion Utility

This section contains samples of the configuration files shown under **"e*Way Configuration File Conversion Utility" on page 23**. Refer to this sample to gain an understanding of the contents of the original files and that of the converted e*Gate 4.5.3 e*Way files.

Figure 6 on page 23 shows a diagram of how the e*Way Configuration File Conversion utility operates. The rest of this section shows examples of the contents of key files shown in the previous figure.

D.2.1 db_send.sc

This section shows an example of the e*Gate 3.6 e*Way's **db_send.sc** file, the original **.sc** file. The following example shows this file before it is converted by the utility:

```
;------------------------------------------------------------------

  ;                General Info
  ; ------------------------------------------------------------------
  -------
  (general-info
    (version "DG-3.6")
    (revision "$Revision: 3.6.2.1 $")
    (user "$Author: cchen $")
    (modified "$Date: 1998/12/19 02:39:52 $")
    (creation "initial")
    (description "ScDb super communication client:

      High level functionality:

      o May be configured as a remote client
  ")
    (user-comment "")
    (generated-cfg-path "configs\ScDb\db_send.cfg")
    (delim1 '\n')
    (delim2 '|')
    (delim3 '=')
    (delim4 ',')
    (cfg-icon "")
  )
  ; ------------------------------------------------------------------
  -------
  ;                DataGateWay Type
  ; ------------------------------------------------------------------
  -------
  (super-client-type
    (network-protocol "<ANY>")
    (os-platform "<ANY>")
    (protocol-api-version "<ANY>")
    (app-protocol "<ANY>")
    (direction "BiDirectional")
  )
  ; ------------------------------------------------------------------
  -------
  ;   Section:"General Settings"
  ; ------------------------------------------------------------------
  -------
```

```
(section
  (name "General Settings")
  (string-set-multi
    (name "Debug Flags")
    (value ())
    (config-default ())
    (set
      (value const ("CHILD" "DBGLD" "IDDBG" "IFCER" "IFCIN" "IFCWA"
"MONKDEBUG" "PROTO" "QUEUE" "RDDTA" "RDMSG" "ROUTE" "RTRSR" "RWDTA"
"SLECT" "TMOUT" "VBOSE" "VERIF" "WTDTA" "WTMSG"))
     (config-default ("CHILD" "DBGLD" "IDDBG" "IFCER" "IFCIN" "IFCWA"
"MONKDEBUG" "PROTO" "QUEUE" "RDDTA" "RDMSG" "ROUTE" "RTRSR" "RWDTA"
"SLECT" "TMOUT" "VBOSE" "VERIF" "WTDTA" "WTMSG"))
    )
    (description "Debug Flags:

The significance of this parameter
is to override the value of the debug
flags as supplied to the communication
client via the -c option. If no options
are selected then the value set by the
-c option takes precedence.

CHILD:   General communication client debug messages
DBGLD:   Configuration file loading information
IDDBG:   Message Identification debug messages
IFCER:   Interface ERROR   debug messages
IFCIN:   Interface INFO    debug messages
IFCWA:   Interface WARNING debug messages
MONKDEBUG: Enables debugging in Monk Scripts
PROTO:   Protocol related debug messages
       (between DataGate server and the super client)
QUEUE:   Queue related debug messages
RDDTA:   General Read operation debug messages
RDMSG:   Message Read operation debug messages
ROUTE:     Routing specific debug message (not used -- has no effect)
RTRSR:     Pending Read debug messages    (not used -- has no effect)
RWDTA:     General Read/Write operation debug messages
SLECT:     Select() call debug messages
TMOUT:     Read/Write Timeout debug messages
VBOSE:     Very verbose Read operation debug messages
       (character by character) -- should be used
        sparingly)
VERIF:     Message verification debug message (not used -- has no
effect)
WTDTA:   General Write operation debug messages
       (with respect to system level calls)
WTMSG:   Message Write operation debug messages
       (with respect to Message/Data sending operations)
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Interface Name")
    (value "ScDb")
    (config-default "ScDb")
    (set
      (value ("ScDb"))
      (config-default ("ScDb"))
    )
    (description "Interface Name:
```

```
The value of \"Interface Name\" is used
to communicate a more verbose name to the
\"status\" commands that originate from
\"dgcmd\" or the DataGate Monitor.

")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "General Settings:

This section contains a set of top level parameters:

    o Debug Flags
    o Interface Name

For communication clients that are local to the DataGate
server running, you do NOT need to set the
\"DataGate server host name\" or the
\"DataGate server port number\" to any value.
")
  (user-comment
    (value "")
    (config-default "")
  )
)
; ------------------------------------------------------------------------
-------
;   Section:"Alert Notification"
; ------------------------------------------------------------------------
-------
(section
  (name "Alert Notification")
  (string-set
    (name "New Control Broker Host Name")
    (value none)
    (config-default none)
    (set
      (value ())
      (config-default ())
    )
    (description "New Control Broker Host Name:

This parameter is used for Alert Notification purposes. Alert
notifications
are done via New Control Broker (newcb). This parameter specifies the
name
of the machine on which newcb is running.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "New Control Broker Port Number")
    (value none)
    (config-default none)
    (set
      (value ())
      (config-default ())
```

```
    )
    (range
      (value (const 2000 const 32767))
      (config-default (2000 32767))
    )
    (description "New Control Broker Port Number:

This parameter is used for Alert Notification purposes. Alert
notifications
are done via New Control Broker (newcb). This parameter specifies the
TCP/IP
port number that newcb is listening for incoming connetcions.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Client ID")
    (value none)
    (config-default none)
    (set
      (value ())
      (config-default ())
    )
    (range
      (value (const 0 const 2147483647))
      (config-default (0 2147483647))
    )
    (description "Client ID:

This parameter is used for Alert Notification purposes. Alert
notifications
are done via New Control Broker (newcb). All DataGateWays must
register with
newcb before they can send out any alert notifications. This parameter
will
be used as a unique identifier for reqistering with newcb.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "Alert Notification:

This section contains the following parameters:

        o New Control Broker Host Name
        o New Control Broker Port Number
        o Client ID

The parameters in this section are used for alert notifications.
")
  (user-comment
    (value "")
    (config-default "")
  )
)
; ------------------------------------------------------------------------
-------
;   Section:"Remote Client Setup"
```

```
; ------------------------------------------------------------------------
-------
(section
  (name "Remote Client Setup")
  (string-set
    (name "Communication Client Location")
    (value "Local")
    (config-default "Remote")
    (set
      (value const ("Local" "Remote"))
      (config-default ("Local" "Remote"))
    )
    (description "Communication Client Location:

This communication client can be setup to run either
\"Local\" to the DataGate server or on a \"Remote\"
machine which is capable of TCP/IP communication
with the outside world (specifically with the
host on which DataGate server will be running on).

To setup this communication client remotely,
you need to follow the procedures in the
DataGate manual. But briefly, a remote
communication client needs the following
as a minimum:

     In its $DATAGATE environment, create
     the log/, queue/, data/, configs/,
     tables/, and any other local directories.

     Make sure that the remote machine
     can see the DataGate server's host.

     Make sure you have also installed
     the proper startup scripts for
     the remote client to start.

See also: \"DataGate Server Host Name\"
     and  \"DataGate Server Port Number\"
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "DataGate Server Host Name")
    (value "localhost")
    (config-default "localhost")
    (set
      (value ("127.0.0.1" "local" "localhost"))
      (config-default ("127.0.0.1" "local" "localhost"))
    )
    (description "DataGate Server Host Name:
(for remote clients only).

This parameter should be set if the
\"Communication Client Location\" parameter
is set to \"Remote\".  The value of this
parameter is the hostname on which the DataGate
server will be running. You may also provide
the IP address of the DataGate host.

When starting this communication client remotely,
```

```
it will either use the value of this parameter
or the value as passed by the -h option to
indicate which host the DataGate Server is
running on. The -h option takes precedence.

*See also: \"DataGate Server Port Number\"
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "DataGate Server Port Number")
    (value 2000)
    (config-default 2000)
    (set
      (value (2000))
      (config-default (2000))
    )
    (range
      (value (const 2000 const 32767))
      (config-default (2000 32767))
    )
    (description "DataGate Server Port Number:

This parameter should be set if the \"Communication
Client Location\" parameter is set to \"Remote\".
The value of this parameter is the port number to
which the DataGate server listens for connections
from communication clients.

When starting this communication client remotely,
it will either use the value of this parameter
or the value as passed by the -p option to
indicate to which port the DataGate Server is
listening for connections. The -p option
takes precedence.

This parameter must be used in conjunction with
\"DataGate Server Host Name\"
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "Remote Client Setup:

The parameters in this section help you set this communication
client as a remote client which will connect and communicate with
the DataGate server on a different machine. The following
is the list of parameters in this section:

    o Communication Client Location (Local or Remote)
    o DataGate Server Host Name
    o DataGate Server Port Number
")
  (user-comment
    (value "")
    (config-default "")
  )
)
```

```
; ------------------------------------------------------------------------
-------
;   Section:"Monk Configuration"
; ------------------------------------------------------------------------
-------
(section
  (name "Monk Configuration")
  (path-set
    (name "Monk File Name")
    (value "tables/dart_send.dgt_dgt/db_send.dsc")
    (config-default none)
    (set
      (value ("tables/dart_send.dgt_dgt/db_send.dsc"))
      (config-default ())
    )
    (description "Monk File Name:

This is the name of the main Monk file containing the
Monk functions to be used by the communication client.
It can appear either as a full pathname to a file, or
as a pathname relative to the $DATAGATE directory.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Main Interface Function")
    (value "db_send")
    (config-default none)
    (set
      (value ("db_send"))
      (config-default ())
    )
    (description "Main Interface Function:

This is the Monk function that will be responsible for either
polling the DataBase or sending information to the DataBase.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Connection Initiation Function")
    (value "db-startup")
    (config-default none)
    (set
      (value ("db-startup"))
      (config-default ())
    )
    (description "Connection Initiation Function:

This Monk function establishes the initial connection to
the DataBase.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
```

```
    (name "Connection Re-Establishment Function")
    (value "db-retry-conn")
    (config-default none)
    (set
      (value ("db-retry-conn"))
      (config-default ())
    )
    (description "Connection Re-Establishment Function:

This Monk function is called repeatedly whenever the
connection to the DataBase is down, and attempts to
re-establish the connection.  If successful, the function
will return UP.  If not, it will return DOWN.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Connection Verification Function")
    (value "db-verify-conn")
    (config-default none)
    (set
      (value ("db-verify-conn"))
      (config-default ())
    )
    (description "Connection Verification Function:

This function is called repeatedly whenever the connection
to the DataBase is thought to be up, and either confirms
that it is still up or discovers that it has gone down.
The function returns either UP or DOWN as appropriate.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Connection Shutdown Function")
    (value "db-shutdown")
    (config-default none)
    (set
      (value ("db-shutdown"))
      (config-default ())
    )
    (description "Connection Shutdown Function:

This Monk function is called when the communication client
shuts down.  It closes the connection to the DataBase.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "DataBase Positive Acknowledgment Function")
    (value "on-send-ack")
    (config-default none)
    (set
      (value ("on-send-ack"))
      (config-default ())
```

```
    )
    (description "DataBase Positive Acknowledgment Function:

This Monk function is responsible for Sending
a positive acknowledgment to the DataBase.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "DataBase Negative Acknowledgment Function")
    (value "on-send-nack")
    (config-default none)
    (set
      (value ("on-send-nack"))
      (config-default ())
    )
    (description "DataBase Negative Acknowledgment Function:

This Monk function is responsible for Sending
a positive acknowledgment to the DataBase.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "DataGate Shutdown Notification Function")
    (value "on-send-shutdown")
    (config-default none)
    (set
      (value ("on-send-shutdown"))
      (config-default ())
    )
    (description "DataGate Shutdown Notification Function:

This Monk function is responsible for Sending
a DGP_SHUTDOWN notice to the DataBase.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "Monk Configuration

The parameters in this section help you set up the required
information for the communication client to utilize Monk.

The parameters in this section are:

    o Monk File Name
    o Main Interface Function
    o Connection Initiation Function
    o Connection Re-Establishment Function
    o Connection Verification Function
    o Connection Shutdown Function
")
  (user-comment
    (value "")
    (config-default "")
```

```
  )
)
; ---------------------------------------------------------------------
-------
;   Section:"DataBase Setup"
; ---------------------------------------------------------------------
-------
(section
  (name "DataBase Setup")
  (string-set
    (name "Inherent Behavior")
    (value "SEND")
    (config-default "SEND")
    (set
      (value const ("POLL" "RECEIVE" "SEND"))
      (config-default ("POLL" "RECEIVE" "SEND"))
    )
    (description "Inherent Behavior:

This parameter allows you to choose whether the communication
client will poll the DataBase or send information to it.
When Inherent Behavior is POLL, this communication client
retrieves data from the DataBase and sends it on to the DataGate
server.  When Inherent Behavior is SEND or RECEIVE, the client
receives
data from DataGate and inserts this data into the DataBase (or uses it
to update existing data in the DataBase). The difference between SEND
and RECEIVE modes of operation is that SEND mode will forward the
reponses generated by the DataBase to DataGate. Receive mode will not
forward anything to DataGate.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Host Name")
    (value "helios8")
    (config-default none)
    (set
      (value ("dart" "helios8"))
      (config-default ())
    )
    (description "Host Name:
If using ODBC: This is the data source name corresponding to the
DataBase.
Otherwise, this is the name of the host machine where the DataBase
resides.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "User Name")
    (value "dgdb")
    (config-default none)
    (set
      (value ("dgdb" "username"))
      (config-default ())
    )
    (description "User Name:
```

```
This is the DataBase user ID.  It is used to
open communication with the DataBase, for both POLL behavior
and SEND behavior.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string encrypt
    (name "Encrypted Password")
    (value "04C8208000")
    (config-default "")
    (description "Encrypted Password:

This is the encrypted DataBase user password.  It is used to
open communication with the DataBase, for both POLL behavior
and SEND behavior.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Down Timeout")
    (value 20)
    (config-default 20)
    (set
      (value (20))
      (config-default (20))
    )
    (range
      (value (const 1 const 86400))
      (config-default (1 86400))
    )
    (description "Down Timeout:

This is the number of seconds to wait between calls to the
Connection Re-Establishment Function.")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Up Timeout")
    (value 20)
    (config-default 20)
    (set
      (value (20))
      (config-default (20))
    )
    (range
      (value (const 1 const 86400))
      (config-default (1 86400))
    )
    (description "Up Timeout:

This is the number of seconds to wait between calls to the
Connection Verification Function.
")
    (user-comment
```

```
        (value "")
        (config-default "")
      )
    )
    (int-set
      (name "Poll Timeout")
      (value 20)
      (config-default 20)
      (set
        (value (20))
        (config-default (20))
      )
      (range
        (value (const 1 const 86400))
        (config-default (1 86400))
      )
      (description "Poll Timeout:

When the communication client's inherent behavior is POLL,
this parameter is the number of seconds to wait between attempts
to poll the DataBase.  This paramater is not used when the
client is configured for SEND behavior.
")
      (user-comment
        (value "")
        (config-default "")
      )
    )
    (int-set
      (name "Resend Timeout")
      (value 20)
      (config-default 20)
      (set
        (value (20))
        (config-default (20))
      )
      (range
        (value (const 1 const 86400))
        (config-default (1 86400))
      )
      (description "Resend Timeout:

When the communication client's inherent behavior is SEND,
this parameter is the number of seconds to wait between attempts
to resend a message to the DataBase (after receiving an error
message from the DataBase).  This paramater is not used when the
client is configured for POLL behavior.
")
      (user-comment
        (value "")
        (config-default "")
      )
    )
    (int-set
      (name "Max Resends Per Message")
      (value 5)
      (config-default 5)
      (set
        (value (5))
        (config-default (5))
      )
      (range
        (value (const 1 const 1000000))
        (config-default (1 1000000))
```

```
    )
    (description "Max Resends Per Message:

This parameter is the maximum number of times the communication
client will attempt to resend a message to the DataBase after
receiving a DataBase error.  When this maximum is reached, the
message is considered a failed message and is written to a journal
file.  This parameter is only used when the inherent behavior of
the client is SEND.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Max Failed Messages")
    (value 3)
    (config-default 3)
    (set
      (value (3))
      (config-default (3))
    )
    (range
      (value (const 1 const 1000000))
      (config-default (1 1000000))
    )
    (description "Max Failed Messages:

This parameter is the maximum number of failed messages the
communication client will allow.  If this many messages fail
and are journaled, the client will shutdown and exit.  As with
the preceding parameter, this parameter is only used for SEND
behavior.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Journal File Name")
    (value "log/db.reject")
    (config-default none)
    (set
      (value ("log/db.reject"))
      (config-default ())
    )
    (description "Journal File Name:

This is the name of the journal file which will be used for storing
failed messages.  The file name may be specified either as a full
pathname to a file, or as a pathname relative to the $DATAGATE
directory.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Zero Wait Between Successful Polls")
    (value "NO")
    (config-default "NO")
```

```
    (set
      (value const ("NO" "YES"))
      (config-default ("NO" "YES"))
    )
    (description "Zero Wait Between Successful Polls:

If this parameter is set to YES then the communication
client will immediately poll the DataBase if the previous
poll returned with data. If this parameter is set to NO
then the client will always wait \"Poll Timeout\" seconds
between polls.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Send Errors to DataGate")
    (value "YES")
    (config-default "YES")
    (set
      (value const ("NO" "YES"))
      (config-default ("NO" "YES"))
    )
    (description "Send Errors to DataGate:

If this parameter is set to YES then error messages received from
the DataBase will be passed on to DataGate.  If this parameter is
set to NO then error messages will not be passed on.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "DataBase Setup

The parameters in this section help you set up the necessary
information for the communication client to be able to connect
and log into the DataBase, to control the behavior of the
communication client in terms of being a polling client or a
sending client, and to set the different timeout values and
recourse logic action to be taken by the communication client.

The parameters in this section are:

    o Inherent Behavior
    o Host Name
    o User Name
    o Encrypted Password
    o Down Timeout
    o Up Timeout
    o Poll Timeout
    o Resend Timeout
    o Max Resends Per Message
    o Max Failed Messages
    o Journal File Name
    o Zero Wait Between Successful Polls
    o Send Errors to DataGate

")
  (user-comment
    (value "")
```

```
    (config-default "")
  )
)
```

D.2.2 dartRule.txt

This file can be appended by using the syntax found in Line 2 with the rules that follow. Insert you new rule in the appropriate section/parameter.

The following example shows the database access rules file:

```
#
#OldSectionName| OldParameterName| NewSectionName| NewParameterName;
Rule1, Rule2,...Rulen
#
#the Rules can be of the following:
#
#IGNORE Will ignore this line
#
#VALUE_REPLACE Replace the value of the new parameter with the value
found
#              from the old parameter. The value is also appended to
the set values
#               if the new parameter is of type Set
#
#SET_APPENDAppend the set values to the new parameter with the set
values
#             found from the old parameter.
#
#SET_REPLACE Replaces the set values to the new parameter. the
current value
#       selected of the new parameter does not change. however, if
the value selected
#       is not part of the set values, then it will be appended to
the set values
#
#POLL_SEND This is a special rule specifically for the old parameter
'Monk File Name'
#       Based on the 'DataBase Setup/Inherent Behavior' value:
#       If 'POLL', replace value of the new parameter 'Process
Outgoing Message Function'
#       else, replace value of the new parameter 'Exchange Data With
External Function'
#       with the value found in the old parameter 'Monk File Name'.
Before value
#       is replaced, it is modified to point to the proper directory
'monk_scripts\common'
#
#if there is no rule defined, section and/or parameter will be copied
over to the new model


General Settings| Debug Flags||;IGNORE
General Settings| Interface Name||;IGNORE
Alert Notification|||;IGNORE
Remote Client Setup|||;IGNORE
Monk Configuration| Monk File Name||;POLL_SEND
Monk Configuration| Main Interface Function|||;IGNORE
Monk Configuration| Connection Initiation Function| Monk
Configuration| Startup Function;VALUE_REPLACE
Monk Configuration| Connection Re-Establishment Function| Monk
Configuration| External Connection Establishment
Function;VALUE_REPLACE
```

```
Monk Configuration| Connection Verification Function| Monk
Configuration| External Connection Verification
Function;VALUE_REPLACE
Monk Configuration| Connection Shutdown Function| Monk Configuration|
External Connection Shutdown Function;VALUE_REPLACE
Monk Configuration| DataBase Positive Acknowledgment Function| Monk
Configuration| Positive Acknowledgment Function;VALUE_REPLACE
Monk Configuration| DataBase Negative Acknowledgment Function| Monk
Configuration| Negative Acknowledgment Function;VALUE_REPLACE
Monk Configuration| DataGate Shutdown Notification Function| Monk
Configuration| Shutdown Command Notification Function;VALUE_REPLACE
DataBase Setup| Host Name| Database Setup|Database Name;VALUE_REPLACE
DataBase Setup| User Name| Database Setup|User Name;VALUE_REPLACE

#
#some old .sc have 'Encrypted Password' and others just have
'Password'
#
DataBase Setup| Encrypted Password| Database Setup| Encrypted
Password;VALUE_REPLACE
DataBase Setup| Password| Database Setup|Encrypted
Password;VALUE_REPLACE
```

D.2.3 dart.def

The following example shows the e*Gate 4.5.3 database access e*Way definition file:

```
; ---------------------------------------------------------------------
-------
;                General Info
; ---------------------------------------------------------------------
-------
(general-info
    (version "eGate")
    (revision "$Revision: 1.1.2.3 $")
    (user "$Author: cchen $")
    (modified "$Date: 2000/03/03 04:20:51 $")
    (creation "initial")
    (description "Database Access Generic e*Way:

    High level functionality:

        o Monk capable

    For more information see the documentation provided with
    this e*Way.
    ")

    (user-comment "")
    (generated-cfg-path "")
    (delim1 '\n')
    (delim2 '|')
    (delim3 '=')
    (delim4 ',')
    (cfg-icon "")
)
; ---------------------------------------------------------------------
-------
;                 e*Way Type
; ---------------------------------------------------------------------
-------
(super-client-type
    (network-protocol     "<ANY>")
```

```
    (os-platform          "<ANY>")
    (protocol-api-version "<ANY>")
    (app-protocol         "<ANY>")
    (direction            "<ANY>")
)
;
;
;-----------------------------------------------------------
;           Section: "General Settings"
;-----------------------------------------------------------
(section
  (name "General Settings")
    (string-set
        (name "Journal File Name")
        (value "")
        (config-default "")
        (set
            (value (""))
            (config-default (""))
        )
        (description "
 Journal File is used for the following conditions:
 - Journal a message when it exceeds the number of retries.
 - Journal an external error when it's not configured to
   forward to Egate.

 If an absolute path is not specified,  the system data
 directory is prepended to the path.
")
        (user-comment
            (value "")
            (config-default "")
        )
  )
  (int-set
    (name "Max Resends Per Message")
    (value 5)
    (config-default 5)
    (set
      (value (5))
      (config-default (5))
    )
    (range
      (value (const 1 const 1024))
      (config-default (1 1024))
    )
    (description "Max Resends Per Message:

 This parameter is the maximum number of times the e*Way
 will attempt to resend a message to the extenal after
 receiving an error.  When this maximum is reached, the
 message is considered a failed message and is written to
 a journal file.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Max Failed Messages")
    (value 3)
    (config-default 3)
    (set
```

```
      (value (3))
      (config-default (3))
    )
    (range
      (value (const 1 const 1024))
      (config-default (1 1024))
    )
    (description "Max Failed Messages:

 This parameter is the maximum number of failed messages
 the e*Way will allow.  If this many messages fail
 and are journaled, the e*Way will shutdown and exit.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Forward External Errors")
    (value const "NO")
    (config-default "NO")
    (set
     (value const ("YES" "NO"))
     (config-default ("YES" "NO"))
    )
    (description "Forward External Errors:

 If this parameter is set to YES then error messages that
 starts with DATAERR received from the external will be
 queued to the configured queue.  If this parameter is set
 to NO then error messages will not be forward.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "General Settings:

 This section contains a set of top level parameters:

        o Journal File Name
        o Max Resends Per Message
        o Max Failed Messages
        o Forward External Errors
")
  (user-comment
    (value "")
    (config-default "")
  )
)
; ------------------------------------------------------------------------
-------
;       Section:        "Communication Setup"
; ------------------------------------------------------------------------
-------
(section
  (name "Communication Setup")
  (schedule-set-multi
    (name "Start Exchange Data Schedule" )
    (value () )
    (config-default () )
    (set
```

```
      (value ())
      (config-default ())
    )
    (description "Start Exchange Data Schedule:

 It either can contain a set of date/time bases schedules,
 or a single repeating (e.g. every n seconds) timer.
 The time instructs the e*Way to exchange data with external
 system.  This set of schedule is used to invoke
 Exchange Data With External Function.  Since Months do not
 fall on even boundaries, it is not provided as a unit.

" )
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (schedule-set-multi
    (name "Stop Exchange Data Schedule" )
    (value () )
    (config-default () )
    (set
      (value ())
      (config-default ())
    )
    (description "Stop Exhange Data Schedule:

 This parameter contains one stop exchange data schedule.
 It either can contain a set of date/time bases schedules,
 or a single repeating (e.g. every n seconds) timer.
 This set of schedule instructs e*Way to stop exchange
 event with external.  Since Months do not fall on even
 boundaries, it is not provided as a unit.

" )
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Exchange Data Interval")
    (value 120)
    (config-default 120)
    (set
      (value (120))
      (config-default (120))
    )
    (range
      (value (const 1 const 86400))
      (config-default (1 86400))
    )
    (description "Exchange Data Interval:
 This is the number of seconds to wait between exchanging data
 attempts.  If \"Zero Wait Between Successful Exchanges\" is
 set to YES and \"Exchange Data with External Function\"
 returns data, this flag is ignored and e*Way will invoke
 \"Exchange Data with External Function\" immediately.
")
    (user-comment
      (value "")
      (config-default "")
    )
```

```
    )
   (int-set
     (name "Down Timeout")
     (value 15)
     (config-default 15)
     (set
       (value (15))
       (config-default (15))
     )
     (range
       (value (const 1 const 86400))
       (config-default (1 86400))
     )
     (description "Down Timeout:

  This is the number of seconds to wait between calls to the
  Connection Establishment Function.")
     (user-comment
       (value "")
       (config-default "")
     )
   )
   (int-set
     (name "Up Timeout")
     (value 15)
     (config-default 15)
     (set
       (value (15))
       (config-default (15))
     )
     (range
       (value (const 1 const 86400))
       (config-default (1 86400))
     )
     (description "Up Timeout:

  This is the number of seconds to wait between calls to the
  Connection Verification Function to verify the connection is
  still up.
")
     (user-comment
       (value "")
       (config-default "")
     )
   )
   (int-set
     (name "Resend Timeout")
     (value 10)
     (config-default 10)
     (set
       (value (10))
       (config-default (10))
     )
     (range
       (value (const 1 const 86400))
       (config-default (1 86400))
     )
     (description "Resend Timeout:

  This parameter is the number of seconds to wait between
  attempts to resend a message to the external (after
  receiving an error message from the external).
")
     (user-comment
```

```
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Zero Wait Between Successful Exchanges")
    (value const "NO")
    (config-default "NO")
    (set
     (value const ("YES" "NO"))
     (config-default ("YES" "NO"))
    )
    (description "Zero Wait Between Successful Exchanges:

 If this parameter is set to YES then the e*Way
 will immediately invoke exchange data with external
 function if the previous exchange function returned
 data. If this parameter is set to NO
 then the e*Way will always wait \"Exchange Data Interval\"
 seconds between invoking exchage data function.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "Communication Setup:

 The parameters in this section help you set this communication
 to external.  The following is the list of parameters in this
section:

        o Start Exchange Data schedule
        o Stop Exchange Data schedule
        o Exchange Data Interval
        o Down Timeout
        o Up Timeout
        o Resend Timeout
        o Zero Wait Between Successful Exchanges

")
  (user-comment
    (value "")
    (config-default "")
  )
)
; ------------------------------------------------------------------
-------
;       Section:        "Monk Configuration"
; ------------------------------------------------------------------
-------
(section
  (name "Monk Configuration")
  (path-set
    (name "Additional Path")
    (value none)
    (config-default none)
    (set
     (value ())
     (config-default ())
    )
    (description "Additional Path:

 Generic e*Way will use bin and shared data paths which
```

```
 comes from egate.store for load path.  On top of
 this, it also uses a path from monk environment
 initialization file.  If this additional path is set,
 it will be appended to the path as well.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Auxiliary Library Directories")
    (value "monk_library/dart")
    (config-default "monk_library/dart")
    (set
     (value ("monk_library/dart"))
     (config-default ("monk_library/dart"))
    )
    (description "Auxiliary Library Directories

 In addition to loading the Monk engine, this e*Way loads
 all the auxiliary library directories if it is configured.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Monk Environment Initialization File")
    (value "db-stdver-init")
    (config-default "db-stdver-init")
    (set
     (value ("db-stdver-init"))
     (config-default ("db-stdver-init"))
    )
    (description "Monk Environment Initialization File:

 This monk environment initialization file will be loaded
 and executed after the auxiliary library directories are
 loaded.  Typically,  It is a good place to initialize any
 global Monk variables that may be used by any other Monk
 Extension Scripts.  e*way will exit if fails to invoke
 this function or this function returns a FAILURE string.
 For example: if it's configured as init_scripts.monk
              init_scripts.monk is loaded and init_scripts
              is invoked.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Startup Function")
    (value "db-stdver-startup")
    (config-default "db-stdver-startup")
    (set
     (value ("db-stdver-startup"))
     (config-default ("db-stdver-startup"))
    )
    (description "Startup Function:

 The Startup Function is invoked by generic e*Way at
```

```
 a startup time or  when configuration changes before
 it enters into its initial Communication State. This
 function is used so that the external system can be
 initialized before message exchange starts.
 This function is called after generic e*Way loads \"Monk
 Environment Initialization File \" and \"Auxiliary directories\".
 e*way will exit if fails to invoke this function or this
 function returns a FAILURE string.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Process Outgoing Message Function")
    (value "db-stdver-proc-outgoing")
    (config-default "db-stdver-proc-outgoing")
    (set
     (value ("db-stdver-proc-outgoing" "db-stdver-proc-outgoing-
stub"))
     (config-default ("db-stdver-proc-outgoing" "db-stdver-proc-
outgoing-stub"))
    )
    (description "Process Outgoing Message Function:

 This function will be loaded and invoked once during the
 initialization process.   e*way will exit if fails to
 invoke this function or this function returns a FAILURE string.
 For example: if it's configured as processOutgoingMsgFunc.monk
              processOutgoingMsgFunc.monk is loaded and
              procesOutgoingMsgFunc is invoked.

 This is the Monk function that will be responsible for
 outgoing event from e*Way the external.  When an e*Way
 has event to send to external, it will invoke this function.
 The function could pass back null string, event, RESEND,
 or error.  Currently, e*Way will only recognize two types
 of errors (conection and data errrors).  Data error string
 must start with \"DATAERR\" string and connection error
 must start with \"CONNERR\".   The returned string that
 starts either with DATAERR or CONNERR will cause a rollback.
 The returned string that starts with DATAERR and extra info
 will be queued to e*Gate if an inbound collaboration is
 configured and if to forward external error is set to YES,
 it will be queued to the configured queue without sending
 ACK/NAK to external.  This parameter is required by e*Way
 so if it's not in configuration file, e*Way will exit.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Exchange Data With External Function")
    (value "db-stdver-data-exchg")
    (config-default "db-stdver-data-exchg")
    (set
     (value ("db-stdver-data-exchg" "db-stdver-data-exchg-stub"))
     (config-default ("db-stdver-data-exchg" "db-stdver-data-exchg-
stub"))
    )
    (description "Exchange Data With External Function:
```

```
 This function will be loaded and invoked once during the
 initialization process.   e*way will exit if fails to
 invoke this function or this function returns a FAILURE string.
 For example: if it's configured as exchangeDataFunc.monk
              exchangeDataFunc.monk is loaded and exchangeDataFunc
              is invoked.

 This function is then invoked at the set schedule and
 will be responsible for either sending or receiving
 data with external.  If this function returns data,
 it will be queued up.  When using this function to
 send out the accumulated data in MONK environment,
 it should return a NULL string.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "External Connection Establishment Function")
    (value "db-stdver-conn-estab")
    (config-default "db-stdver-conn-estab")
    (set
     (value ("db-stdver-conn-estab"))
     (config-default ("db-stdver-conn-estab"))
    )
    (description "External Connection Establishment Function:

 This Monk function is called repeatedly at the set
 interval whenever the connection to the external is
 down or unknown state, and attempts to establish the
 connection.  If successful, the function will return
 UP.  If not, it will return DOWN.  A string is neither
 UP nor DOWN will result e*Way in external unknown state.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "External Connection Verification Function")
    (value "db-stdver-conn-ver")
    (config-default "db-stdver-conn-ver")
    (set
     (value ("db-stdver-conn-ver"))
     (config-default ("db-stdver-conn-ver"))
    )
    (description "External Connection Verification Function:

 This Monk function is called repeatedly at the set
 interval whenever the connection to the external is
 thought to be up, and either confirms that it is
 still up or discovers that it has gone down.  The
 function returns either UP or DOWN as appropriate.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
```

```
    (name "External Connection Shutdown Function")
    (value "db-stdver-conn-shutdown")
    (config-default "db-stdver-conn-shutdown")
    (set
     (value ("db-stdver-conn-shutdown"))
     (config-default ("db-stdver-conn-shutdown"))
    )
    (description "External Connection Shutdown Function:

 This Monk function is called when the e*Way
 shuts down.  It can be used to clean up before
 e*Way exits.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Positive Acknowledgment Function")
    (value "db-stdver-pos-ack")
    (config-default "db-stdver-pos-ack")
    (set
     (value ("db-stdver-pos-ack"))
     (config-default ("db-stdver-pos-ack"))
    )
    (description "Positive Acknowledgment Function:

 This function will be loaded and invoked once during the
 initialization process.   e*way will exit if fails to
 invoke this function or this function returns a FAILURE string.
 For example: if it's configured as ackFunc.monk
              ackFunc.monk is loaded and ackFunc is invoked.

  This Monk function is called when the e*Way
 successfully processes and queues data from external.
 This function can return data to be queued but e*Way
 won't ACK/NAK on the data.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Negative Acknowledgment Function")
    (value "db-stdver-neg-ack")
    (config-default "db-stdver-neg-ack")
    (set
     (value ("db-stdver-neg-ack"))
     (config-default ("db-stdver-neg-ack"))
    )
    (description "Negative Acknowledgment Function:

 This function will be loaded and invoked once during the
 initialization process.   e*way will exit if fails to
 invoke this function or this function returns a FAILURE string.
 For example: if it's configured as nackFunc.monk
              nackFunc.monk is loaded and nackFunc is invoked.

 This Monk function is called when the e*Way
 fails to process and queue data from external.
 This function can return data to be queued but e*Way
 won't ACK/NAK on the data.
```

```
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Shutdown Command Notification Function")
    (value "db-stdver-shutdown")
    (config-default "db-stdver-shutdown")
    (set
     (value ("db-stdver-shutdown"))
     (config-default ("db-stdver-shutdown"))
    )
    (description "Shutdown Command Notification Function:

 This Monk function is called when an e*Way needs
 to shutdown, it notifies the external that e*Way
 is about to shutdown.  This function can be used
 to shutdown connection with external.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "Monk Configuration

 The parameters in this section help you set up the
 required information for the e*Way to utilize Monk.

 The parameters in this section are:

        o Additional Path
        o Auxiliary Library Directories
        o Monk Environment Initialization File
        o Startup Function
        o Process Outgoing Message Function
        o Exchange Data With External Function
        o External Connection Establishment Function
        o External Connection Verification Function
        o External Connection Shutdown Function
        o Positive Acknowledgment Function
        o Negative Acknowledgment Function
        o Shutdown Command Notification Function
")
  (user-comment
    (value "")
    (config-default "")
  )
)
;-------------------------------------------------------------
;           Section: "Database Setup"
;-------------------------------------------------------------
(section
  (name "Database Setup")
  (string-set
    (name "Database Type")
    (value "SYBASE")
    (config-default "SYBASE")
    (set
     (value ("SYBASE" "ORACLE7" "ORACLE8" "ORACLE8i" "ODBC"))
     (config-default ("SYBASE" "ORACLE7" "ORACLE8" "ORACLE8i" "ODBC"))
    )
```

```
    (description "Database Type:

This is the type of the database.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )

  (string-set
    (name "Database Name")
    (value none)
    (config-default none)
    (set
     (value ())
     (config-default ())
    )
    (description "Database Name:

This is the name of the database.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )  (string-set
    (name "User Name")
    (value none)
    (config-default none)
    (set
     (value ())
     (config-default ())
    )
    (description "User Name:

 This is the user name of the database.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string encrypt
    (name "Encrypted Password")
    (value "")
    (config-default "")
    (description "Encrypted Password:

 This is the password of the database.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "Database Setup:

 This section contains a set of top level parameters:

        o Database Type
        o Database Name
        o User name
        o Encrypted Password
```

```
")
  (user-comment
    (value "")
    (config-default "")
  )
)
```

5.2.7 db_send.sc

The following example shows the new **.sc** file for the e*Gate 4.5.3 e*Way:

```
; ---------------------------------------------------------------------
-------
;                General Info
; ---------------------------------------------------------------------
-------
(general-info
  (version "eGate")
  (revision "$Revision: 1.1.2.3 $")
  (user "$Author: cchen $")
  (modified "$Date: 2000/03/03 04:20:51 $")
  (creation "initial")
  (description "Database Access Generic e*Way:

    High level functionality:

        o Monk capable

    For more information see the documentation provided with
    this e*Way.
    ")
  (user-comment "")
  (generated-cfg-path "configs/stcewgenericmonk/db_send.cfg")
  (delim1 '\n')
  (delim2 '|')
  (delim3 '=')
  (delim4 ',')
  (cfg-icon "")
)
; ---------------------------------------------------------------------
-------
;                 E*Way Type
; ---------------------------------------------------------------------
-------
(super-client-type
  (network-protocol "<ANY>")
  (os-platform "<ANY>")
  (protocol-api-version "<ANY>")
  (app-protocol "<ANY>")
  (direction "BiDirectional")
)
; ---------------------------------------------------------------------
-------
;   Section:"General Settings"
; ---------------------------------------------------------------------
-------
(section
  (name "General Settings")
  (string-set
    (name "Journal File Name")
    (value "")
    (config-default "")
    (set
```

```
      (value (""))
      (config-default (""))
    )
    (description "
 Journal File is used for the following conditions:
 - Journal a message when it exceeds the number of retries.
 - Journal an external error when it's not configured to
   forward to Egate.

 If an absolute path is not specified,  the system data
 directory is prepended to the path.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Max Resends Per Message")
    (value 5)
    (config-default 5)
    (set
      (value (5))
      (config-default (5))
    )
    (range
      (value (const 1 const 1024))
      (config-default (1 1024))
    )
    (description "Max Resends Per Message:

 This parameter is the maximum number of times the e*Way
 will attempt to resend a message to the extenal after
 receiving an error.  When this maximum is reached, the
 message is considered a failed message and is written to
 a journal file.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Max Failed Messages")
    (value 3)
    (config-default 3)
    (set
      (value (3))
      (config-default (3))
    )
    (range
      (value (const 1 const 1024))
      (config-default (1 1024))
    )
    (description "Max Failed Messages:

 This parameter is the maximum number of failed messages
 the e*Way will allow.  If this many messages fail
 and are journaled, the e*Way will shutdown and exit.
")
    (user-comment
      (value "")
      (config-default "")
    )
```

```
  )
  (string-set
    (name "Forward External Errors")
    (value "NO")
    (config-default "NO")
    (set
      (value const ("NO" "YES"))
      (config-default ("NO" "YES"))
    )
    (description "Forward External Errors:

 If this parameter is set to YES then error messages that
 starts with DATAERR received from the external will be
 queued to the configured queue.  If this parameter is set
 to NO then error messages will not be forward.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "General Settings:

 This section contains a set of top level parameters:

        o Journal File Name
        o Max Resends Per Message
        o Max Failed Messages
        o Forward External Errors
")
  (user-comment
    (value "")
    (config-default "")
  )
)
; ------------------------------------------------------------------------
-------
;   Section:"Communication Setup"
; ------------------------------------------------------------------------
-------
(section
  (name "Communication Setup")
  (schedule-set-multi
    (name "Start Exchange Data Schedule")
    (value ())
    (config-default ())
    (set
      (value ())
      (config-default ())
    )
    (description "Start Exchange Data Schedule:

 It either can contain a set of date/time bases schedules,
 or a single repeating (e.g. every n seconds) timer.
 The time instructs the e*Way to exchange data with external
 system.  This set of schedule is used to invoke
 Exchange Data With External Function.  Since Months do not
 fall on even boundaries, it is not provided as a unit.

")
    (user-comment
      (value "")
      (config-default "")
    )
```

```
  )
  (schedule-set-multi
    (name "Stop Exchange Data Schedule")
    (value ())
    (config-default ())
    (set
      (value ())
      (config-default ())
    )
    (description "Stop Exhange Data Schedule:

 This parameter contains one stop exchange data schedule.
 It either can contain a set of date/time bases schedules,
 or a single repeating (e.g. every n seconds) timer.
 This set of schedule instructs e*Way to stop exchange
 event with external.  Since Months do not fall on even
 boundaries, it is not provided as a unit.

")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Exchange Data Interval")
    (value 120)
    (config-default 120)
    (set
      (value (120))
      (config-default (120))
    )
    (range
      (value (const 1 const 86400))
      (config-default (1 86400))
    )
    (description "Exchange Data Interval:
 This is the number of seconds to wait between exchanging data
 attempts.  If \"Zero Wait Between Successful Exchanges\" is
 set to YES and \"Exchange Data with External Function\"
 returns data, this flag is ignored and e*Way will invoke
 \"Exchange Data with External Function\" immediately.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Down Timeout")
    (value 15)
    (config-default 15)
    (set
      (value (15))
      (config-default (15))
    )
    (range
      (value (const 1 const 86400))
      (config-default (1 86400))
    )
    (description "Down Timeout:

 This is the number of seconds to wait between calls to the
 Connection Establishment Function.")
```

```
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Up Timeout")
    (value 15)
    (config-default 15)
    (set
      (value (15))
      (config-default (15))
    )
    (range
      (value (const 1 const 86400))
      (config-default (1 86400))
    )
    (description "Up Timeout:

 This is the number of seconds to wait between calls to the
 Connection Verification Function to verify the connection is
 still up.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (int-set
    (name "Resend Timeout")
    (value 10)
    (config-default 10)
    (set
      (value (10))
      (config-default (10))
    )
    (range
      (value (const 1 const 86400))
      (config-default (1 86400))
    )
    (description "Resend Timeout:

 This parameter is the number of seconds to wait between
 attempts to resend a message to the external (after
 receiving an error message from the external).
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Zero Wait Between Successful Exchanges")
    (value "NO")
    (config-default "NO")
    (set
      (value const ("NO" "YES"))
      (config-default ("NO" "YES"))
    )
    (description "Zero Wait Between Successful Exchanges:

 If this parameter is set to YES then the e*Way
 will immediately invoke exchange data with external
 function if the previous exchange function returned
```

```
 data. If this parameter is set to NO
 then the e*Way will always wait \"Exchange Data Interval\"
 seconds between invoking exchage data function.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "Communication Setup:

 The parameters in this section help you set this communication
 to external.  The following is the list of parameters in this
section:

        o Start Exchange Data schedule
        o Stop Exchange Data schedule
        o Exchange Data Interval
        o Down Timeout
        o Up Timeout
        o Resend Timeout
        o Zero Wait Between Successful Exchanges

")
  (user-comment
    (value "")
    (config-default "")
  )
)
; ---------------------------------------------------------------------
-------
;   Section:"Monk Configuration"
; ---------------------------------------------------------------------
-------
(section
  (name "Monk Configuration")
  (path-set
    (name "Additional Path")
    (value none)
    (config-default none)
    (set
      (value ())
      (config-default ())
    )
    (description "Additional Path:

 Generic e*Way will use bin and shared data paths which
 comes from egate.store for load path.  On top of
 this, it also uses a path from monk environment
 initialization file.  If this additional path is set,
 it will be appended to the path as well.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Auxiliary Library Directories")
    (value "monk_library/dart")
    (config-default "monk_library/dart")
    (set
      (value ("monk_library/dart"))
      (config-default ("monk_library/dart"))
```

```
    )
    (description "Auxiliary Library Directories

 In addition to loading the Monk engine, this e*Way loads
 all the auxiliary library directories if it is configured.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Monk Environment Initialization File")
    (value "db-stdver-init")
    (config-default "db-stdver-init")
    (set
      (value ("db-stdver-init"))
      (config-default ("db-stdver-init"))
    )
    (description "Monk Environment Initialization File:

 This monk environment initialization file will be loaded
 and executed after the auxiliary library directories are
 loaded.  Typically,  It is a good place to initialize any
 global Monk variables that may be used by any other Monk
 Extension Scripts.  e*way will exit if fails to invoke
 this function or this function returns a FAILURE string.
 For example: if it's configured as init_scripts.monk
              init_scripts.monk is loaded and init_scripts
              is invoked.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Startup Function")
    (value "db-startup")
    (config-default "db-stdver-startup")
    (set
      (value ("db-startup" "db-stdver-startup"))
      (config-default ("db-stdver-startup"))
    )
    (description "Startup Function:

 The Startup Function is invoked by generic e*Way at
 a startup time or  when configuration changes before
 it enters into its initial Communication State. This
 function is used so that the external system can be
 initialized before message exchange starts.
 This function is called after generic e*Way loads \"Monk
 Environment Initialization File \" and \"Auxiliary directories\".
 e*way will exit if fails to invoke this function or this
 function returns a FAILURE string.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Process Outgoing Message Function")
    (value "monk_scripts/common/db_send.dsc")
```

```
    (config-default "db-stdver-proc-outgoing")
    (set
      (value ("db-stdver-proc-outgoing" "db-stdver-proc-outgoing-
stub" "monk_scripts/common/db_send.dsc"))
      (config-default ("db-stdver-proc-outgoing" "db-stdver-proc-
outgoing-stub"))
    )
    (description "Process Outgoing Message Function:

 This function will be loaded and invoked once during the
 initialization process.   e*way will exit if fails to
 invoke this function or this function returns a FAILURE string.
 For example: if it's configured as processOutgoingMsgFunc.monk
              processOutgoingMsgFunc.monk is loaded and
              procesOutgoingMsgFunc is invoked.

 This is the Monk function that will be responsible for
 outgoing event from e*Way the external.  When an e*Way
 has event to send to external, it will invoke this function.
 The function could pass back null string, event, RESEND,
 or error.  Currently, e*Way will only recognize two types
 of errors (conection and data errrors).  Data error string
 must start with \"DATAERR\" string and connection error
 must start with \"CONNERR\".   The returned string that
 starts either with DATAERR or CONNERR will cause a rollback.
 The returned string that starts with DATAERR and extra info
 will be queued to e*Gate if an inbound collaboration is
 configured and if to forward external error is set to YES,
 it will be queued to the configured queue without sending
 ACK/NAK to external.  This parameter is required by e*Way
 so if it's not in configuration file, e*Way will exit.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Exchange Data With External Function")
    (value "db-stdver-data-exchg")
    (config-default "db-stdver-data-exchg")
    (set
      (value ("db-stdver-data-exchg" "db-stdver-data-exchg-stub"))
      (config-default ("db-stdver-data-exchg" "db-stdver-data-exchg-
stub"))
    )
    (description "Exchange Data With External Function:

 This function will be loaded and invoked once during the
 initialization process.   e*way will exit if fails to
 invoke this function or this function returns a FAILURE string.
 For example: if it's configured as exchangeDataFunc.monk
              exchangeDataFunc.monk is loaded and exchangeDataFunc
              is invoked.

 This function is then invoked at the set schedule and
 will be responsible for either sending or receiving
 data with external.  If this function returns data,
 it will be queued up.  When using this function to
 send out the accumulated data in MONK environment,
 it should return a NULL string.
")
    (user-comment
      (value "")
```

```
        (config-default "")
      )
    )
    (string-set
      (name "External Connection Establishment Function")
      (value "db-retry-conn")
      (config-default "db-stdver-conn-estab")
      (set
        (value ("db-retry-conn" "db-stdver-conn-estab"))
        (config-default ("db-stdver-conn-estab"))
      )
      (description "External Connection Establishment Function:

 This Monk function is called repeatedly at the set
 interval whenever the connection to the external is
 down or unknown state, and attempts to establish the
 connection.  If successful, the function will return
 UP.  If not, it will return DOWN.  A string is neither
 UP nor DOWN will result e*Way in external unknown state.
")
      (user-comment
        (value "")
        (config-default "")
      )
    )
    (string-set
      (name "External Connection Verification Function")
      (value "db-verify-conn")
      (config-default "db-stdver-conn-ver")
      (set
        (value ("db-stdver-conn-ver" "db-verify-conn"))
        (config-default ("db-stdver-conn-ver"))
      )
      (description "External Connection Verification Function:

 This Monk function is called repeatedly at the set
 interval whenever the connection to the external is
 thought to be up, and either confirms that it is
 still up or discovers that it has gone down.  The
 function returns either UP or DOWN as appropriate.
")
      (user-comment
        (value "")
        (config-default "")
      )
    )
    (string-set
      (name "External Connection Shutdown Function")
      (value "db-shutdown")
      (config-default "db-stdver-conn-shutdown")
      (set
        (value ("db-shutdown" "db-stdver-conn-shutdown"))
        (config-default ("db-stdver-conn-shutdown"))
      )
      (description "External Connection Shutdown Function:

 This Monk function is called when the e*Way
 shuts down.  It can be used to clean up before
 e*Way exits.
")
      (user-comment
        (value "")
        (config-default "")
      )
```

```
  )
  (path-set
    (name "Positive Acknowledgment Function")
    (value "on-send-ack")
    (config-default "db-stdver-pos-ack")
    (set
      (value ("db-stdver-pos-ack" "on-send-ack"))
      (config-default ("db-stdver-pos-ack"))
    )
    (description "Positive Acknowledgment Function:

 This function will be loaded and invoked once during the
 initialization process.   e*way will exit if fails to
 invoke this function or this function returns a FAILURE string.
 For example: if it's configured as ackFunc.monk
              ackFunc.monk is loaded and ackFunc is invoked.

  This Monk function is called when the e*Way
 successfully processes and queues data from external.
 This function can return data to be queued but e*Way
 won't ACK/NAK on the data.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (path-set
    (name "Negative Acknowledgment Function")
    (value "on-send-nack")
    (config-default "db-stdver-neg-ack")
    (set
      (value ("db-stdver-neg-ack" "on-send-nack"))
      (config-default ("db-stdver-neg-ack"))
    )
    (description "Negative Acknowledgment Function:

 This function will be loaded and invoked once during the
 initialization process.   e*way will exit if fails to
 invoke this function or this function returns a FAILURE string.
 For example: if it's configured as nackFunc.monk
              nackFunc.monk is loaded and nackFunc is invoked.

 This Monk function is called when the e*Way
 fails to process and queue data from external.
 This function can return data to be queued but e*Way
 won't ACK/NAK on the data.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Shutdown Command Notification Function")
    (value "on-send-shutdown")
    (config-default "db-stdver-shutdown")
    (set
      (value ("db-stdver-shutdown" "on-send-shutdown"))
      (config-default ("db-stdver-shutdown"))
    )
    (description "Shutdown Command Notification Function:

 This Monk function is called when an e*Way needs
```

```
 to shutdown, it notifies the external that e*Way
 is about to shutdown.  This function can be used
 to shutdown connection with external.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "Monk Configuration

 The parameters in this section help you set up the
 required information for the e*Way to utilize Monk.

 The parameters in this section are:

        o Additional Path
        o Auxiliary Library Directories
        o Monk Environment Initialization File
        o Startup Function
        o Process Outgoing Message Function
        o Exchange Data With External Function
        o External Connection Establishment Function
        o External Connection Verification Function
        o External Connection Shutdown Function
        o Positive Acknowledgment Function
        o Negative Acknowledgment Function
        o Shutdown Command Notification Function
")
  (user-comment
    (value "")
    (config-default "")
  )
)
; ------------------------------------------------------------------------
-------
;   Section:"Database Setup"
; ------------------------------------------------------------------------
-------
(section
  (name "Database Setup")
  (string-set
    (name "Database Type")
    (value "SYBASE")
    (config-default "SYBASE")
    (set
      (value ("ODBC" "ORACLE7" "ORACLE8" "ORACLE8i" "SYBASE"))
      (config-default ("ODBC" "ORACLE7" "ORACLE8" "ORACLE8i"
"SYBASE"))
    )
    (description "Database Type:

This is the type of the database.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "Database Name")
    (value "helios8")
    (config-default none)
    (set
```

```
      (value ("helios8"))
      (config-default ())
    )
    (description "Database Name:

This is the name of the database.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string-set
    (name "User Name")
    (value "dgdb")
    (config-default none)
    (set
      (value ("dgdb"))
      (config-default ())
    )
    (description "User Name:

 This is the user name of the database.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (string encrypt
    (name "Encrypted Password")
    (value "04C8208000")
    (config-default "")
    (description "Encrypted Password:

 This is the password of the database.
")
    (user-comment
      (value "")
      (config-default "")
    )
  )
  (description "Database Setup:

 This section contains a set of top level parameters:

        o Database Type
        o Database Name
        o User name
        o Encrypted Password
")
  (user-comment
    (value "")
    (config-default "")
  )
)
db_send.cfg (New .cfg file e*Gate 4.5.3 e*Way)
#
# ------------------------------------------------------------------------
-------
#               Delimiters To Use
# ------------------------------------------------------------------------
-------
#
```

```
File/CFG/Version:0.0/Delim1:\o012/Delim2:\o174/Delim3:\o075/
Delim4:\o054
#
#
# ------------------------------------------------------------------------
-------
#                General Info
# ------------------------------------------------------------------------
-------
#
#  version:eGate
#  revision:$Revision: 1.1.2.3 $
#  user      :$Author: cchen $
#  modified:$Date: 2000/03/03 04:20:51 $
#  creation:initial
#
#
# ------------------------------------------------------------------------
-------
#                 E*Way Type
# ------------------------------------------------------------------------
-------
#
#  network-protocol:<ANY>
#  os-platform:<ANY>
#  protocol-api-version:<ANY>
#  app-protocol:<ANY>
#  direction:BiDirectional
#
#
# ------------------------------------------------------------------------
-------
#   Section:General Settings
# ------------------------------------------------------------------------
-------
#
General Settings|Journal File Name|value=|set=
General Settings|Max Resends Per Message|value=5|set=5|range=1,1024
General Settings|Max Failed Messages|value=3|set=3|range=1,1024
General Settings|Forward External Errors|value=NO|set=NO,YES
#
# ------------------------------------------------------------------------
-------
#   Section:Communication Setup
# ------------------------------------------------------------------------
-------
#
Communication Setup|Start Exchange Data Schedule|value=|set=
Communication Setup|Stop Exchange Data Schedule|value=|set=
Communication Setup|Exchange Data
Interval|value=120|set=120|range=1,86400
Communication Setup|Down Timeout|value=15|set=15|range=1,86400
Communication Setup|Up Timeout|value=15|set=15|range=1,86400
Communication Setup|Resend Timeout|value=10|set=10|range=1,86400
Communication Setup|Zero Wait Between Successful
Exchanges|value=NO|set=NO,YES
#
# ------------------------------------------------------------------------
-------
#   Section:Monk Configuration
# ------------------------------------------------------------------------
-------
#
Monk Configuration|Additional Path|value=|set=
```

```
Monk Configuration|Auxiliary Library Directories|value=monk_library/
dart|set=monk_library/dart
Monk Configuration|Monk Environment Initialization File|value=db-
stdver-init|set=db-stdver-init
Monk Configuration|Startup Function|value=db-startup|set=db-
startup,db-stdver-startup
Monk Configuration|Process Outgoing Message
Function|value=monk_scripts/common/db_send.dsc|set=db-stdver-proc-
outgoing,db-stdver-proc-outgoing-stub,monk_scripts/common/db_send.dsc
Monk Configuration|Exchange Data With External Function|value=db-
stdver-data-exchg|set=db-stdver-data-exchg,db-stdver-data-exchg-stub
Monk Configuration|External Connection Establishment
Function|value=db-retry-conn|set=db-retry-conn,db-stdver-conn-estab
Monk Configuration|External Connection Verification
Function|value=db-verify-conn|set=db-stdver-conn-ver,db-verify-conn
Monk Configuration|External Connection Shutdown Function|value=db-
shutdown|set=db-shutdown,db-stdver-conn-shutdown
Monk Configuration|Positive Acknowledgment Function|value=on-send-
ack|set=db-stdver-pos-ack,on-send-ack
Monk Configuration|Negative Acknowledgment Function|value=on-send-
nack|set=db-stdver-neg-ack,on-send-nack
Monk Configuration|Shutdown Command Notification Function|value=on-
send-shutdown|set=db-stdver-shutdown,on-send-shutdown
#
# ------------------------------------------------------------------
-------
#   Section:Database Setup
# ------------------------------------------------------------------
-------
#
Database Setup|Database
Type|value=SYBASE|set=ODBC,ORACLE7,ORACLE8,ORACLE8i,SYBASE
Database Setup|Database Name|value=helios8|set=helios8
Database Setup|User Name|value=dgdb|set=dgdb
Database Setup|Encrypted Password|value=04C8208000
```

D.2.4 db_send.ctl

The following example shows the control (**.ctl**) file created for the e*Gate 4.5.3 e*Way:

```
db_send.cfg,configs/stcewgenericmonk,FILETYPE_ASCIITEXT
db_send.sc,configs/stcewgenericmonk,FILETYPE_ASCIITEXT
```

D.2.5 db_send.sc.old

This is the original file displayed in **"db_send.sc" on page 88** renamed with the **.old** extension.

Glossary

Access Control List (ACL)
The security feature in e*Gate; a role-based list of information that specifies which users have permission to access e*Gate and its components and what specific access rights the users have.

advisory lock
The lock placed on a file when a user checks it out from the run-time schema. An advisory lock is simply a flag that warns other users that someone is already editing the file; it does not prevent other users from also checking out the file.

agent (Alert, SNMP)
A stand-alone application that monitors processes and resources and sends Notifications to e*Gate system users, informing them of system status (for example, when a preset disk space level is exceeded).

Application Programing Interface (API)
An API is the set of classes, functions, and methods of a particular programming language that developers use to code software. API documentation is the documenting of the syntax and use of the API methods.

business Event
A unit of data sent by an external system to e*Gate representing a change in that system's information.

Business Object Broker (BOB)
The executable component **stcbob.exe**. BOBs use Collaborations to route and transform data within the e*Gate system.

Business Rules pane
Use the **Business Rules** pane in the Java Collaboration Rules Editor to navigate and edit the Java code of a Collaboration.

Business Rules toolbar
Use the buttons on the Business Rules toolbar in the Java Collaboration Rules Editor to add corresponding Java statements to a Collaboration.

byte length
Length in bytes of the string or regular expression to be matched within an Event Type Definition. e*Gate measures fixed-length data from byte 1.

byte offset
The beginning byte location of the string or regular expression to be verified within an Event Type Definition, beginning at byte 0.

child nodes
Nodes that are below a given node within the same branch of the Event Type Definition tree. Child nodes can inherit certain properties, such as delimiters, from their parent nodes.

Collaboration
The component within an e*Way or BOB that performs data transformation and/or routing. It is the business logic that is applied to an Event in the course of delivery from a publisher to a subscriber. Collaboration components do the following functions: Subscriber components receive Events of a known type while publisher components distribute the transformed Events to a specified recipient. See also **Collaboration Rules**.

Collaboration Rules Editors
The graphical user interface (GUI) features used to work with Collaboration Rules scripts in the Java and Monk programming languages. See also **Collaboration Rules script.**

Collaboration-ID Rules Editor
The graphical user interface (GUI) feature used to create Collaboration Rules scripts in the Monk programming language for *e*Gate release 3.6 only*. See also **Collaboration Rules script.**

Collaboration Rules
The program logic that instructs a Collaboration how to execute the business logic required to support e*Gate's data transformation and routing. See also **Collaboration** and **Collaboration script**.

Collaboration Rules script
A Collaboration script (data program) written using the Collaboration Rules Editor feature.

Collaboration script
The data flow and transformation logic contained in and configured by an e*Gate Collaboration and written as a program in any of the following programming languages: Monk, Java, or C.

Collaboration Services
Libraries that provide the low-level facilities by which Collaborations execute Collaboration Rules, for example, issuing system-specific terminate calls.

command line
A tool for monitoring and controlling e*Gate by entering application program interface (API) commands at a DOS or DOS-type prompt.

committing files
Takes them out of the run-time schema and places them in the Sandbox. See also, **Sandbox** and **run time**.

Control Broker
An automatically generated e*Gate component that starts and monitors e*Ways and BOBs. At least one Control Broker must be running on each host within a schema.

delimiter
A special character assigned to mark the boundary of an Event node.

delimiter declaration field
In the HL7 standard, the location within an Event where a character is to be used as a delimiter. Also refers to the Event Type Definition node boundary it marks.

destination
Pertaining to the primary output Event Type Definition within a Collaboration Rules component or Collaboration Rules script.

e*Gate Monitor
A standard e*Gate component that provides graphical access to e*Gate systems and e*Gate status information, state control, and troubleshooting log files and journals.

e*Way Connection
An e*Way Connection is the encoding of the access information for one particular external connection or SeeBeyond JMS IQ Manager. In terms of content, it is similar to an e*Way configuration file, in defining enough information to be able to "login" or connect to the particular system. However, unlike e*Way configuration files, there is no schedule information. The idea is that the e*Way Connection will be information shared across multiple interfaces.

e*Way Configuration Editor
The graphical user interface (GUI) feature used to configure e*Ways.

e*Way Intelligent Adapter
A component that provides a noninvasive point of contact between an e*Gate system and an external business application (often abbreviated as e*Way). e*Ways establish connectivity with applications, using whatever communication protocol is appropriate. e*Ways perform the following main functions: (1) receiving unprocessed data from external components, transforming it into Events, and forwarding it to other components within e*Gate via Intelligent Queues (IQs); and (2) sending processed data to external components (can also include data transformation).

Enterprise Manager
The e*Gate graphical user interface (GUI) that allows you to create, configure, and modify all components of an e*Gate system.

Event
A unit package of data processed by the e*Gate system. This data has a defined structure, for example, a known number of fields with known characteristics and

delimiters. Events are classified by type (Event Type) and exchanged within e*Gate as Event Type Definitions (ETDs).

Event, delimited
A variable-length Event made up of nodes whose boundaries are marked by delimiters.

Event, fixed
An Event of prescribed length. Each node within a fixed Event Type Definition is identified by its length and location within that Event Type Definition.

Event Linking and Sequencing (ELS)
Event Linking and Sequencing is a feature that allows for Events that arrive from independent input streams to be delivered to subscribers as related units. Complex Linking and Sequencing can be configured using the e*Gate 4.5 Java Collaboration Rules Editor, so that **n** different input streams can be linked and sequenced according to rules based on any combination of content or time-out rules.

Event, monitoring
An Event sent from one e*Gate component to another that describes an internal e*Gate condition, such as "component up" or "component down."

Event Type
A class of Events with common data structure (for example, a known number of fields, with known characteristics and delimiters). An Event Type is also a logical name entry in e*Gate that points to a single Event Type Definition (ETD).

Event Type Definition (ETD)
A programmatic representation of an Event Type that Collaboration Rules can use when parsing, transforming, or routing data.

Event Type Definition Editors
The graphical user interface (GUI) features used to configure Event Type Definitions (ETDs) in the Java and Monk programming languages; abbreviated as ETD Editor. See also **Event Type Definition.**

Event Type Definition node
A segment of an Event Type Definition (ETD) that is represented graphically as a node in an Event Type Definition tree in the Event Type Definition Editor window, and represents a portion of an Event.

Event Type Definition tree
The graphical or logical representation of the Event Type Definition and its hierarchy.

Extensible Markup Language (XML)
w3.org defines Extensible Markup Language (XML) as the universal format for structured documents and data on the Web.

external system
A system that sends or receives data and is outside of the e*Gate system.

Guaranteed Exactly Once Delivery (GEOD)
Using XA, GEOD guarantees once and only once delivery. Guaranteed Exactly Once Delivery refers to the usage of XA-compliant e*Gate and external components to ensure the delivery occurs once regardless of failures.

ignore
When a file from the run-time schema, which already carries an advisory lock, is checked out. The advisory lock stays with the original user who checked out the file, and does not transfer to the new user.

instance
A specific node within a series of repeating nodes.

Intelligent Queue (IQ)
A standard e*Gate component that manages the exchange of information between components within the e*Gate system, providing nonvolatile storage for data as it passes from one component to another.

IQ Manager
A standard e*Gate component that reorganizes Intelligent Queues (IQs), archives queue information upon request to save disk space, and locks the queues when maintenance is performed.

IQ Service
A utility that provides the transport of components within Intelligent Queues (IQs), handling the low-level implementation of data exchange, such as system calls to initialize or reorganize a database.

Java Message Service (JMS)
See **SeeBeyond JMS** for the e*Gate implementation of JMS.

log file
A text file that contains a record of all actions taken by an e*Way. Use log files to troubleshoot any problems in the system and discover how to solve them.

Monitor
An executable e*Gate component that enables users to view messages that describe the state of e*Gate internal components. Interactive monitors also enable users to send commands to e*Gate components; non-interactive monitors only enable users to view notifications.

monitoring Event
An Event, sent by one e*Gate component to another (usually to the Control Broker) that describes occurrences within the e*Gate system. Monitoring Events include error messages, such as "component down" or "component lost"; status messages such as "component up" or "contact re-established"; system performance messages, such as "event processing below preset threshold" or "disk space low"; and miscellaneous messages such as scheduled timers, configuration changes, or "event content of interest."

Monk
SeeBeyond's event-processing language.

Monk Test Console
A graphical user interface (GUI) test feature for testing Monk functions and Collaboration scripts before introducing them into the run-time environment.

Navigator Tree
The tree-like graphical display in the Navigator/Components pane of the Enterprise Manager window. This display shows the components of the e*Gate system and how they relate to each other in pictorial form using an icon to represent each component.

node
See **Event Type Definition node**.

node set
A group of associated nodes that are order-independent, or that repeat.

notification
A notification sent to the user by the e*Gate system.

notification routing
The Collaboration Rules script that specifies how monitoring Events are translated into notifications.

parent nodes
Nodes that are above a given node within the same branch of the Event Type Definition tree.

Participating Host
A client computer that supports an e*Gate system, as opposed to the Registry Host, which acts as a server to the Participating Host.

promoting files
Update the run-time schema to use the new file or files. If the file already exists in the run-time schema, that file is replaced with the file from the Sandbox. Promoting a file automatically removes it from the user's Sandbox and, if the user has locked the file, releases the lock. When you delete a file from the Sandbox without promoting it to the run-time schema, you *remove* the file. If the file was locked, the lock is released.

publish
See **publish/subscribe**.

publish/subscribe
Abbreviated as pub/sub; subscriber components retrieve Events. Publisher components make Events available to other e*Gate components. See also **Collaboration**.

Registry
The storage place (in a directory) for all e*Gate configuration details, including file containment.

Registry Service
The service that handles all requests for updates to the e*Gate registry and forwards updated files to clients as necessary.

regular expression
A pattern representing a set of strings to be matched.

removing files
Delete a file from the Sandbox without promoting it to the run-time schema. If the user carried the advisory lock for the file, the lock is released.

root node
The highest-level node in the Event Type Definition tree.

run time
The environment in the Registry shared by all users of that Registry. The run time contains parameters that run for each instance of e*Gate unless the controlling user has a parameter in his or her own Sandbox, in which case the Sandbox is overridden. The run time is the production environment of a schema. See also, **Team Registry**.

Sandbox
A user's local development area. Each user has his own Sandbox. Files in a user's Sandbox are available for testing the functions in the file themselves, but they are not available to the run-time schema. In other words, files within a person's Sandbox are not available to the e*Gate components (such as e*Ways or BOBs) that use them. See also, **Team Registry**.

schema
Includes files and associated stores created by e*Gate, which contain the parameters of all the components that control, route, and transform data as it moves through e*Gate in a predefined system configuration.

SeeBeyond JMS
e*Gate implementation of the Java Message Service (JMS) using IQ Managers, IQs, and a special e*Way Connection.

sibling nodes
Nodes that are children of the same parent node.

source
Pertaining to the primary input Event or Event Type Definition within a Collaboration or Collaboration script.

subnode
A node that is connected through parent-child relationships to another node that is higher in the Event Type Definition tree.

subnode set
A set of order-independent or repeating Event Type Definition nodes one level below the currently selected node in the Event Type Definition tree.

subscribe
See **publish/subscribe**.

Team Registry
Allows multiple users to develop components of a single schema simultaneously by compartmentalizing the e*Gate Registry into work-in-progress and run-time environment areas, implemented by the Sandbox and run-time environments.

Index

A

automate the conversion process **16**

B

Batch e*Way **24**, **40**, **41**, **43**

C

COM/DCOM **24**, **40**, **41**, **43**
COM/DCOM e*Way **40**, **41**, **43**
Compatibility Components **14**, **15**, **16**, **65**
conventions, writing in document **10**
CORBA Visibroker **24**
CORBA Visibroker e*Way **40**, **41**, **43**
creating IQs **20**

D

DART e*Way **24**
DART for Oracle **24**
DART for Sybase **24**
DataGate Collaboration Service **19**
DataGateWay-to-e*Way Utility **23**
debug privilege **67**
dgw2ew.cmd **23**, **40**
document purpose and scope **9**

E

e*Gate Compatibility Components **14**, **15**, **16**, **65**

F

FAQs **64**
    upgrading **66**
FTP **24**
FTP e*Way **40**, **41**, **43**

G

get function **66**

I

Installation **15**
intended audience, document **9**

L

Loop syntax **50**

M

Monk functions
    $standard->julian **50**
    %standard->julian **50**
    upward compatible **66**
MQ-configure-options **62**
MQSeries e*Way **24**, **40**, **41**, **43**

O

ODBC database access e*Way **40**, **41**, **43**
Oracle database access e*Way **40**, **41**, **43**
organization of information, document **10**

P

PeopleSoft Message Agent **24**
PeopleSoft Message Agent e*Way **40**, **41**, **43**
Prerequisites for migration **30**
Proxy e*Way **18**
Proxy e*Ways, creating **18**
publication/subscription **20**

R

RTC Service **16**, **18**, **20**, **21**, **22**, **28**
    role of RTC in schema creation **19**

S

SAP ALE **24**
SAP ALE e*Way **40**, **41**, **43**
SAP BAPI **24**
SAP BAPI e*Way **40**, **41**, **44**
SAP BDC **24**
SAP BDC e*Way **40**, **41**, **44**
SAP EDI **24**
SAP EDI e*Way **40**, **41**, **44**
Schema Configuration Utility **17**
SeeBeyond Web site
    additional information
        technical support **13**
    support **66**
set! function **66**

Siebel EIM **24**
Siebel EIM e*Way **40**, **41**, **44**
Siebel Event Driven **24**
Siebel Event Driven e*Way **41**, **44**
single route **17**
standard DataGateWays **24**
stc_collabdatagate.dll **19**
stcdgschema **28**, **30**, **40**
    configures Proxy e*Ways **18**
    configures RTC Service **19**
    defined **16**
    generates .cfg and .sc files **19**
    -noJavaConnectionPoints
        FIFO capability **27**
    option arguments for **17**
    running defines e*Ways **38**
    running the utility **32**, **33**
    with -singleroute flag **21**, **22**
stcewproxy.exe **18**
support password, obtaining **66**
Supported Versions **16**
supporting documents **12**
Sybase database access e*Way **40**, **41**, **43**
System Requirements **14**

T

TCP/IP HL7 **24**
TCP/IP HL7 e*Way **41**, **44**

U

Undefining the DGOS Environment Variable **37**
upgrading
    Communication Clients **67**
    custom e*Ways **67**
    DataGate to e*Gate **66**
    e*Gate
        from 4.X to 4.5.1 or later **67**
    Monk functions **66**
utf8convert.exe **78**