SeeBeyond ICAN Suite

e*Way Intelligent Adapter for CORBA-VisiBroker (Server) User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

SEEBEYOND®

Version 20050405222501.

Contents

Chapter 1

Introduction

This document provides instructions for installing and configuring the SeeBeyond™ Technology Corporation's (SeeBeyond™) e*Way™ Intelligent Adapter CORBA-VisiBroker (Server). This chapter provides an introduction to the e*Way.

1.1 CORBA-VisiBroker: Overview

The CORBA-VisiBroker Server e*Way enables CORBA-VisiBroker applications to exchange data with non-CORBA systems.

Common Object Request Broker Architecture (CORBA) is a standard for transferring messages between objects (software programs) in a distributed environment. CORBA allows applications written in different programming languages and running on different platforms to communicate with each other and exchange data according to the particular format required by each object.

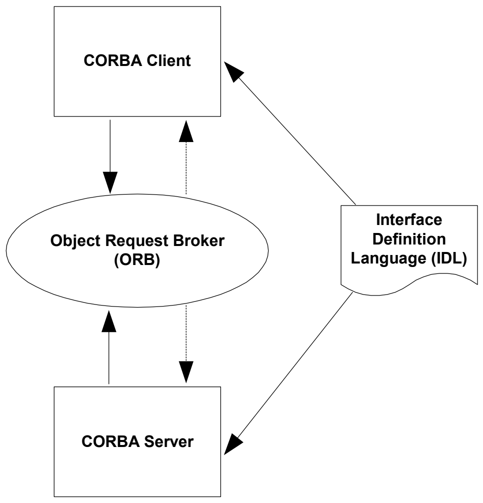
CORBA objects are defined by an Interface Definition Language (IDL), which is then compiled for an object's specific programming language. An IDL describes the methods (processing instructions) for the object and the required format for all data sent and returned by the object.

CORBA client objects make requests to CORBA server objects through an Object Request Broker (ORB). An ORB transparently invokes methods on the server objects, so client objects do not have to be aware of a server object's location, programming language, or other system parameters. An ORB is the middleware that ties remote CORBA applications together.

CORBA clients and servers communicate as defined by the General Inter-ORB Protocol (GIOP). When the GIOP is transmitted using TCP/IP, the network transport is defined as the Internet Inter-ORB Protocol (IIOP).

CORBA architecture exists as a three-tier system, as illustrated in Figure 1.

Figure 1 CORBA Architecture


CORBA Client
Object Request Broker (ORB)
Interface Definition Language (IDL)
CORBA Server

1.1.1 CORBA-VisiBroker Server e*Way

The CORBA-VisiBroker Server e*Way links external CORBA applications with non-CORBA processes, enabling data exchange across an enterprise system that incorporates a variety of different applications. The e*Way functions as a CORBA server in a CORBA environment that uses Inprise's VisiBroker for Java version 3.4, and is compliant with CORBA version 2.1.

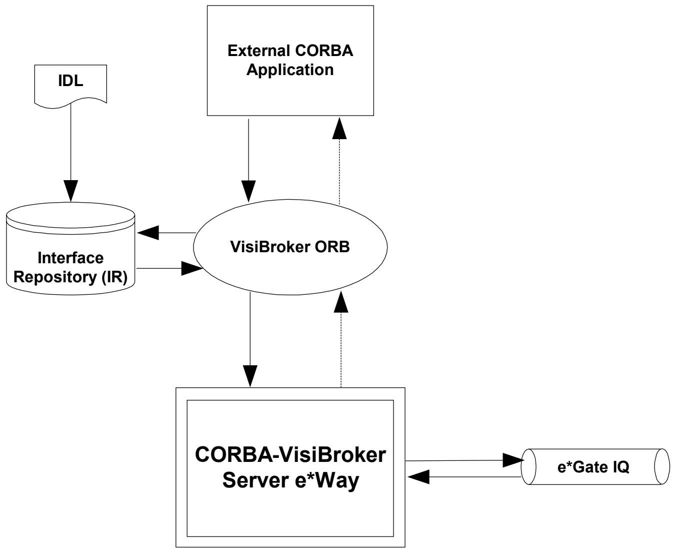
Functional Overview

The CORBA-VisiBroker Server e*Way utilizes the CORBA Dynamic Skeleton Interface (DSI) to dynamically handle client object invocations. IDLs are stored in an Interface Repository (IR), and all objects in the system are registered with the ORB. At runtime the ORB queries the IR to obtain the necessary parameters for a particular object, and the e*Way processes the request.

Note: *The CORBA-VisiBroker Server e*Way handles all incoming and outgoing data dynamically. Handling data in this manner may result in slower system performance as compared with an application hard-coded to a particular IDL. For information on compiling a specific IDL into a custom e*Way, please contact SeeBeyond Customer Service and Support.*

Figure 2 illustrates how the CORBA-VisiBroker Server e*Way fits into a CORBA-VisiBroker enterprise system:

Figure 2 e*Way System Architecture

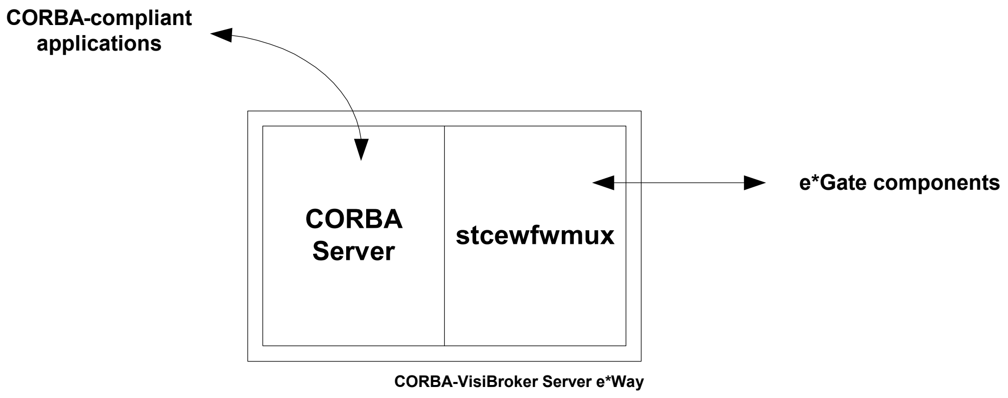

External CORBA Application
IDL
Interface Repository (IR)
VisiBroker ORB
CORBA-VisiBroker Server e*Way
e*Gate IQ

The CORBA-VisiBroker Server e*Way supports multiple, concurrent requests from CORBA clients, providing multi-threaded capability. After the e*Way receives a request, the following steps occur:

1. The request is passed to the e*Gate system as an asynchronous Event.
2. e*Gate processes the Event according to the business logic defined by the user.
3. The response is correlated with the CORBA request and the result is returned to the CORBA client.

The e*Way itself contains two components: The CORBA Server component which communicates with the ORB, and the **stcewfwmux** component, which communicates with other components in the e*Gate system. The following diagram represents these functional parts:

Figure 3 CORBA-VisiBroker Server e*Way – Functional Parts



CORBA Server

The CORBA Server component of the e*Way is a java process that dynamically creates server objects, allowing users to define the IDL e*Gate will implement on the fly. The Server, in conjunction with the ORB, queries the IR at runtime to determine the IDL interfaces to be implemented; is registered with the ORB as a server for the object interfaces; and accepts requests from CORBA client invocations.

The CORBA-VisiBroker Server e*Way handles multiple client requests in a multi-threaded fashion. If the Event backlog reaches a certain point, the e*Way starts multiple copies of the Server process to distribute the workload. This functionality is inherent to the e*Way and does not require user intervention.

stcewfwmux

The stcewfwmux portion of the e*Way is the executable component that communicates directly with other e*Gate components. This portion of the e*Way takes its parameters from a user-defined e*Way configuration file. For information on the configuration file, see **Chapter 3**.

1.1.2 Intended Reader

The reader of this guide is presumed to be a developer or system administrator with responsibility for setting up or maintaining the e*Gate Integrator system. The reader should have high-level knowledge of Windows and UNIX operations and administration, and be thoroughly familiar with all applications within the CORBA-VisiBroker environment at the user's site. Anyone responsible for configuring or monitoring the e*Gate system should also be thoroughly familiar with Windows-style GUI operations.

1.2 Components

The CORBA-VisiBroker Server e*Way comprises the following components:

- **stcewfwmux.exe**, the executable component
- **egate.jar**, the CORBA Server component
- Configuration files, which the e*Way Editor uses to define configuration parameters
- The CORBA VisiBroker Server Converter, a build tool that creates Monk Event Type Definitions from IDL files

A complete list of installed files appears in **Table 1 on page 13**.

1.3 Supported Operating Systems

The CORBA-VisiBroker Server e*Way is available on the following operating systems:

- Windows 2000, Windows XP, and Windows Server 2003
- Sun Solaris 8 and 9

1.4 System Requirements

To use the CORBA-VisiBroker Server e*Way, you need to meet the following requirements:

- An e*Gate Participating Host
- A TCP/IP network connection

  The e*Way must be configured and administered using the e*Gate Schema Designer.

Note: *Additional disk space can be required to process and queue the data that this e*Way processes. The amount necessary can vary based on the type and size of the data being processed and any external applications doing the processing.*

- Pentium-class 866 MHz CPU
- VisiBroker for Java version 3.4, installed on the same machine as the e*Way
- Running VisiBroker interface repository

VisiBroker for Java has its own requirements; see this system's documentation for details.

1.5 External System Requirements

The CORBA-VisiBroker Client e*Way supports the following external system:

- CORBA applications

CORBA applications may have their own requirements; see the specific application's documentation for details.

To enable the e*Way to communicate properly with the external CORBA application, VisiBroker for Java version 3.4 and the e*Way must be run on the same machine. Additionally, you must have a VisiBroker interface repository running on a machine within the CORBA/e*Gate network.

Chapter 2

Installation

This chapter covers how to install the CORBA-VisiBroker Server e*Way. It also includes a list of the files and directories the installation process creates.

2.1 Installation on Windows Systems

2.1.1 Pre-installation

- Exit all Windows programs before running the setup program, including any anti-virus applications.
- You must have Administrator privileges to install this e*Way.

2.1.2 Installation Procedure

To install the CORBA-VisiBroker Server e*Way Windows systems

1. Log in as an Administrator on the workstation on which you want to install the e*Way.
2. Insert the installation CD-ROM into the CD-ROM drive.
3. If the CD-ROM drive's "Autorun" feature is enabled, the setup application should launch automatically; skip ahead to step 4. Otherwise, use Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the file **setup.exe** on the CD-ROM drive.
4. The InstallShield setup application launches. Follow the installation instructions until you come to the **Please choose the product to install** dialog box.
5. Select **e*Gate Integrator**, then click **Next**.
6. Follow the on-screen instructions until you come to the second **Please choose the product to install** dialog box.
7. Clear the check boxes for all selections except **Add-ons**, and then click **Next**.
8. Follow the on-screen instructions until you come to the **Select Components** dialog box.
9. Highlight (but do not check) **e*Ways**, and then click the **Change** button. The **SelectSub-components** dialog box appears.

10 Select the **CORBA-VisiBroker Server e*Way**. Click the continue button to return to the **Select Components** dialog box, then click **Next**.

11 Follow the rest of the on-screen instructions to install the CORBA-VisiBroker Server e*Way. Be sure to install the e*Way files in the suggested client installation directory. The installation utility detects and suggests the appropriate installation directory.

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

12 After the installation is complete, exit the install utility.

13 Copy the file named **stcjcs.jar** from **<*eGate*>\server\registry\repository\classes** to **<*eGate*>\client\classes,** where **<*eGate*>** is the directory in which e*Gate is installed.

Note: *Once you have installed and configured this e*Way, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, IQs, and Event Types before this e*Way can perform its intended functions. For more information about any of these procedures, please see the online Help.*

*For more information about configuring e*Ways or how to use the e*Way Editor, see the* ***e*Gate Integrator User's Guide****.*

2.2 Installation on UNIX Systems

2.2.1 Pre-installation

You do not require root privileges to install this e*Way. Log in under the user name that you wish to own the e*Way files. Be sure that this user has sufficient privilege to create files in the e*Gate directory tree.

2.2.2 Installation Procedure

To install the CORBA-VisiBroker Server e*Way on a UNIX system

1 Log in on the workstation containing the CD-ROM drive, and insert the CD-ROM into the drive.

2 If necessary, mount the CD-ROM drive.

3 At the shell prompt, type

```
cd  /cdrom
```

4 Start the installation script by typing

```
setup.sh
```

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

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

6 After the installation is complete, exit the install utility.

7 Copy the file named **stcjcs.jar** from **<*eGate*>\server\registry\repository\classes** to **<*eGate*>\client\classes,** where **<*eGate*>** is the directory in which e*Gate is installed.

8 Launch the Schema Designer.

Note: *Once you have installed and configured this e*Way, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, IQs, and Event Types before this e*Way can perform its intended functions. For more information about any of these procedures, please see the online Help system.*

*For more information about configuring e*Ways or how to use the e*Way Editor, see the **e*Gate Integrator User's Guide**.*

2.3 Files/Directories Created by the Installation

The CORBA-VisiBroker Server e*Way installation process installs the files shown in Table 1 within the e*Gate directory tree. Files are installed within the **eGate** tree on the Participating Host and committed to the "default" schema on the Registry Host.

Table 1 Files Created by Installation

Directories	Files
client\bin	stcewfwmux.exe
client\configs\stcewfwmux	stcewfwmux.def
client\monk_scripts\common	request.ssc reply.ssc

Chapter 3

Configuration

This chapter describes how to configure the CORBA-VisiBroker Server e*Way.

3.1 e*Way Configuration Parameters

e*Way configuration parameters are set using the e*Way Editor.

To change e*Way configuration parameters:

1. In the Schema Designer's Component editor, select the e*Way you want to configure and display its properties.
2. Under **Configuration File**, click **New** to create a new file, **Find** to select an existing configuration file, or **Edit** to edit the currently selected file.
3. In the **Additional Command Line Arguments** box, type any additional command line arguments that the e*Way may require, taking care to insert them *at the end* of the existing command-line string. Be careful not to change any of the default arguments unless you have a specific need to do so.

For more information about how to use the e*Way Editor, see the e*Way Editor's online Help or the *Working with e*Ways* chapter in the *e*Gate Integrator User's Guide*.

The e*Way's configuration parameters are organized into the following sections:

- Server Settings
- Communication Settings

3.1.1 Server Settings

The parameters in the Server Settings section define information about the Server portion of the e*Way.

Server Name

Description

Specifies the name of the host on which the e*Way will run. ***This parameter is required.***

Required Values

A valid hostname.

Server IP Port

Description

Specifies the port number through which the executable component of the e*Way will communicate with the CORBA Server component.

Required Values

An integer between 2000 and 65536. The default is 27538.

Server Interface Repository Name

Description

The name of the Interface Repository.

Required Values

Any valid name.

Additional Information

The CORBA Server component of the e*Way assumes a default name of EGATEREP. If you choose a different name for the interface repository, you must specify it in this parameter.

Auto Start Server

Description

Specifies whether or not to automatically start the CORBA Server portion of the e*Way when the **stcewfwmux.exe** executable is started.

Required Values

Yes or **No**.

Server Exec String

Description

Specifies the command used to autostart the CORBA Server portion of the e*Way.

Required Values

A valid java command. This parameter is required if the **Auto Start Server** parameter is set to **Yes**.

Caution: ***Do not modify the default value unless you are directed to do so by SeeBeyond support personnel.***

Additional Information

Following are additional arguments you can add to the command line:

Argument	Description	Default
- ObjImplName	The CORBA object implementation name that the CORBA Server portion of the e*Way gives to each of the classes it implements.	eGateObj
- modq	The module qualifier. The e*Way will implement interfaces in all modules in the interface repository that begin with this string.	eGate

Classpath

Description

Specifies the path to the e*Gate and VisiBroker java classes. This is the environment variable CLASSPATH.

Required Values

A valid CLASSPATH. For information on setting the CLASSPATH variable, see **"Setting the CLASSPATH"**, below.

3.1.2 Communication Settings

The parameter in this section controls retry actions for the e*Way's connection to the external system.

Connect Retry Timeout

Description

Specifies the number of seconds to wait between connection attempts.

Required Values

An integer between 1 and 65536.

3.2 Environment Configuration

This section describes changes required to the operating environment to support the CORBA-VisiBroker Server e*Way.

3.2.1 Setting the PATH

In order for the e*Way to function properly you must ensure that the following Java processes are defined in the PATH variable on the machine on which the e*Way is running:

- **current dir**

- **jdk1.1.8/bin**
- **inprise/vbroker_java/bin**

3.2.2 Setting the CLASSPATH

In order for the e*Way to function properly you must ensure that the following Java processes are defined in the CLASSPATH variable on the machine on which the e*Way is running:

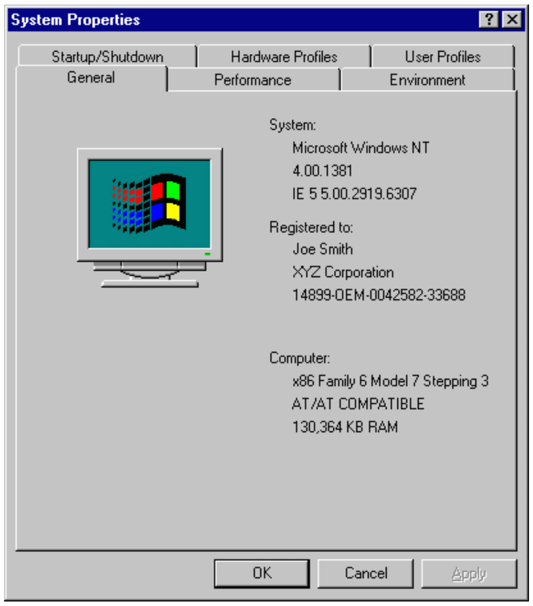
- **egate.jar**
- **vbjapp.jar**
- **vbjorb.jar**
- **vbjtools.jar**
- **swingall.jar**
- **stcjcs.jar**
- **xml4j.jar**
- **. (current directory)**

Windows Systems

To define the CLASSPATH variable on Windows

1. On the taskbar, click the **Start** button, point to **Settings**, then click **Control Panel**.
2. Select **System**.
3. Pull down the **File** menu and select **Open** to open the System Properties dialog box.

Figure 4 System Properties



System:
Microsoft Windows NT
4.00.1381
IE 5 5.00.2919.6307

Registered to:
Joe Smith
XYZ Corporation
14899-OEM-0042582-33688

Computer:
x86 Family 6 Model 7 Stepping 3
AT/AT COMPATIBLE
130,364 KB RAM

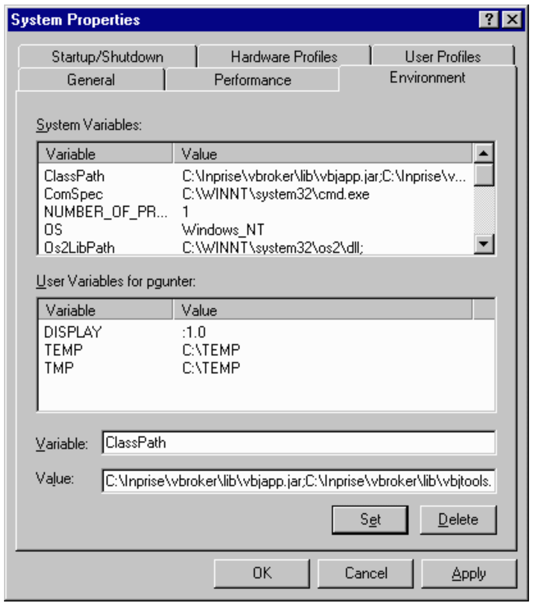
4 Select the **Environment** tab.

5 Under **System Variables** select **ClassPath**.

6 In the **Value** field, type the path to the directories that contain the **.jar** files listed above. Multiple paths should be separated by a semi-colon.

The final result should look something like this:

Figure 5 Windows ClassPath Variable



System Variables:

Variable	Value
ClassPath	C:\Inprise\vbroker\lib\vbjapp.jar;C:\Inprise\v...
ComSpec	C:\WINNT\system32\cmd.exe
NUMBER_OF_PR...	1
OS	Windows_NT
Os2LibPath	C:\WINNT\system32\os2\dll;

User Variables for pgunter:

Variable	Value
DISPLAY	:1.0
TEMP	C:\TEMP
TMP	C:\TEMP

Variable: ClassPath

Value: C:\Inprise\vbroker\lib\vbjapp.jar;C:\Inprise\vbroker\lib\vbjtools.

7 Click **Set** to assign the value to the variable.

8 Click **OK** to close the dialog box.

3.3 External Configuration Requirements

There are no configuration changes required in the external system. All necessary configuration changes can be made within e*Gate.

Chapter 4

Implementation

This chapter explains procedures related to implementing the CORBA-VisiBroker Server e*Way in a production environment. It includes a working example of the e*Way and a list of supported IDL types.

4.1 Using the ETD Editor's Build Tool

The CORBA VisiBroker Server Converter creates an Event Type Definition (ETD) file automatically by interrogating the Interface Repository. The nodes of the ETD are used to both represent details of the interface definition (method name, parameter names/data-types/direction) and hold the argument values passed during invocation.

You can access the build tool directly from the ETD Editor graphical user interface (GUI) or run it from the command line or a command script.

Use one of the following procedures to create an Event Type Definition with the Build tool.

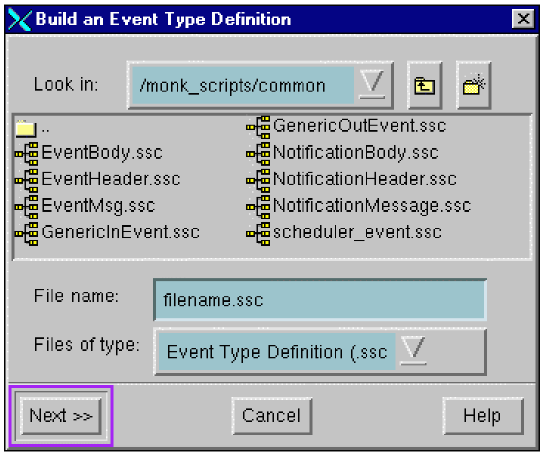
Important: *The Interface Repository must be running before you use the build tool.*

4.1.1 Creating Event Type Definitions from the GUI

To create an ETD using the CORBA VisiBroker Server Converter from the GUI:

1. Launch the ETD Editor.
2. On the ETD Editor's Toolbar, click **Build**. The **Build an Event Type Definition** dialog box appears (see **Figure 6 on page 20**).

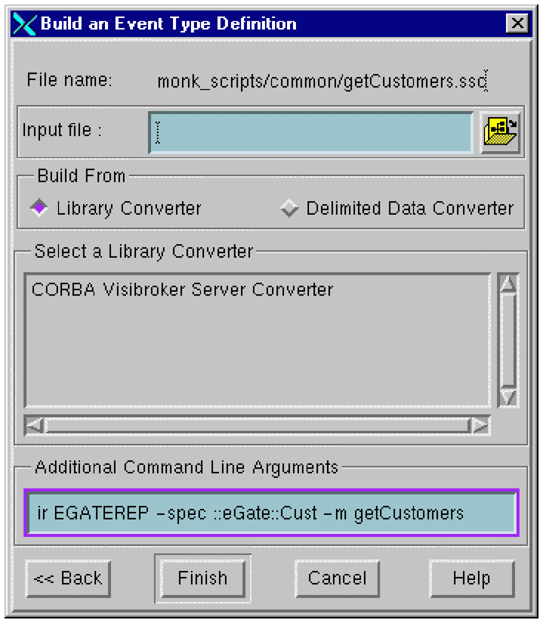
Figure 6 Build an Event Type Definition



3 In the **File name** field, type the name of the ETD file you wish to build. Do not specify any file extension—the Editor will supply the .ssc extension automatically.

4 Click **Next**. A new dialog box displays.

Figure 7 Build an Event Type Definition - CORBA VisiBroker Server Converter


Build an Event Type Definition
File name:
monk_scripts/common/getCustomers.ssc
Input file :
Build From
Library Converter
Delimited Data Converter
Select a Library Converter
CORBA Visibroker Server Converter
Additional Command Line Arguments
ir EGATEREP -spec ::eGate::Cust -m getCustomers
<< Back
Finish
Cancel
Help

5 Leave the **Input file** field blank—the CORBA VisiBroker Server Converter does not use this field.

6 Under **Build From**, select **Library Converter**.

7 Under **Select a Library Converter**, select **CORBA VisiBroker Server Converter**.

8 Under **Additional Command Line Arguments**, type the following:

```
-ir rep_name -spec path -m method_name
```

where

- *rep_name* is the name of the Interface Repository
- *path* is the module/interface path

spec paths are separated with a double colon (::) and always start with leading double colons (::), but do not contain trailing colons. For example:

```
-spec ::eGate::account::balance
```

- *method_name* is the name of the method

9 Click **Finish**. The Build tool will create the ETD.

4.1.2 Creating Event Type Definitions from a Command Line

To create an ETD using the CORBA VisiBroker Server Converter from a command line:

From the command line type the following on one line:

```
jre com.stc.xml.ir2ssc.IR2SSC -ir rep_name -o output_file.ssc -spec
path -m method_name
```

where

- *rep_name* is the name of the Interface Repository
- *output_file* is the name of the ETD being created
- *path* is the module path within the IDL to the interface in which you want to generate the ETD

  spec paths are separated with a double colon (::) and always start with leading double colons (::), but do not contain trailing colons. For example:

  ```
  -spec ::eGate::account::balance
  ```
- *method_name* is the name of the method

Example

The following example creates an Event Type Definition from an IDL.

1 Create a text file with the following IDL code segment:

```
module eGate {
     module account {
          interface balance {
               long get(in NameFormat accountName,
                         in AccountFormat currentAccount);
          };
     };
};
```

2 Save the file as **account.idl**.

3 To load **account.idl** into the Interface Repository **EGATEREP**, type the following on a command line:

```
irep -console EGATEREP account.idl.
```

4 To generate the Event Type Definition for the eGate::account::balance::get() function, execute the following command:

```
jre -classpath %CLASSPATH% com.stc.xml.ir2ssc.IR2SSC -ir EGATEREP
-o bget.ssc -spec ::eGate::account::balance -m get
```

Command Definition for the Converter

Usage:

```
java IR2SSC [-ir name] [-e[rrlog] filename] [-h[elp]]
[-m[ethod] methodName] [-noSSC] [-s[ilent] [-spec
::mod::mod...::interface] [-v[erbose] value]
```

spec paths are "::" separated, always start with a leading "::", and never contain a trailing "::". The following is an example spec path:

```
-spec ::eGate::account::balance
```

The command flags are described in the following table.

Table 2 Converter Command Arguments

Flag	Purpose
-e [rrlog] file	Use this to log all status and error output to the specified log file. The default is to log to STDERR.
-h [elp]	This lists all the parameters for the ir2ssc command. Note: This message (-help) will always go to STDOUT and CLI errors will always go to STDERR. Errors encountered while opening the log file will be reported to STDERR.
-ir repname	Bind to specific IR.
-m [ethod] name	This specifies which method (operation) to extract from the repository. The default is to extract all methods.
-noSSC	This instructs the command not to generate the ETD. (default: false)
-s [ilent]	This instructs the command not to generate any verbose output. (By default, this parameter is set to false). Note: Using the silent parameters always overrides verbose output.
-spec name	This instructs the command to only process operations that match the ::module::interface spec.
-tf file	This is the same as the errlog option.
-v [verbose] val	This sets the verbose ess level. The default setting is 1. Possible values are: -vl Displays major section status -v25 Displays runtime status -v50 Displays flattened XML structure -v75 Displays initial XML (w/references) and flattened XML -v100 Displays more information than is needed

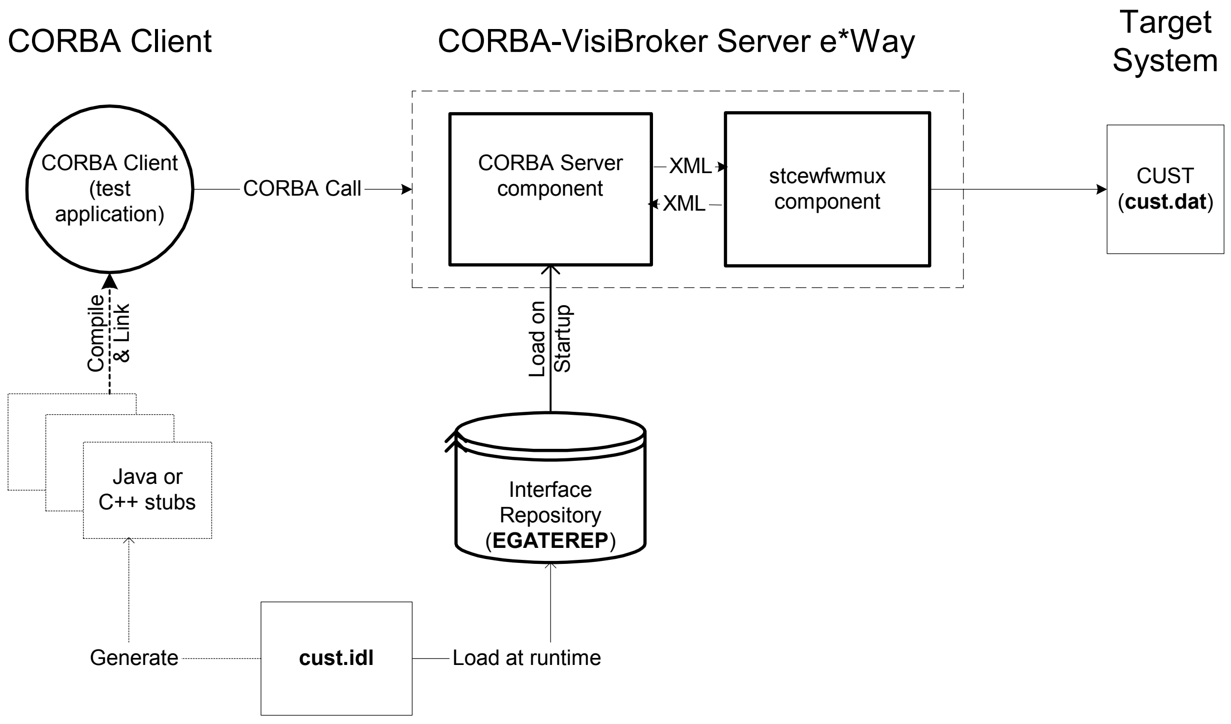
4.2 Working Example

The installation CD-ROM contains a sample implementation of the CORBA-VisiBroker Server e*Way. The CD-ROM contains all of the files required for a complete implementation, including the CORBA client application, the IDL file, the target system, and the e*Gate schema. To complete the example, you must have an Interface Repository available to you. In order for the example to be implemented universally, the target system is emulated within e*Gate. However, the technique can be used for any system that e*Gate connects to, such as databases, COM applications, or SAP.

To run the example, you can use the files on the CD-ROM or you can create all of the necessary components yourself and verify them against the files on the CD-ROM. This chapter includes instructions on how to create the CORBA client application, IDL, and the target system. You can either import the sample schema components directly from the CD-ROM or you can create the schema yourself. For instructions on how to import the sample schema, see **"Importing the Sample Schema" on page 34**. For instructions on how to create the sample schema yourself, see **"Creating the Sample Schema" on page 25**.

The following diagram illustrates the workflow in the example:

Figure 8 CORBA-VisiBroker Server e*Way Example



The implementation has been designed to be as natural to CORBA application developers as possible. A CORBA client application makes a call to a non-CORBA, target system, **CUST**, represented by the file **cust.dat**. Before you run the example, you must load the IDL file **cust.idl** into the Interface Repository (named **EGATEREP** in this example). When the CORBA Server portion of the e*Way starts, it interrogates the

Interface Repository and loads the IDL. When the e*Way receives the CORBA call from the client application, it maps the call to the IDL and converts the data to an XML Event. After processing the Event the e*Way sends the target system's reply back to the CORBA client; in this example, the CORBA client displays the contents of **cust.dat**.

Implementing the example involves five basic steps:

1. Creating the target system (**cust.dat**).
2. Writing the IDL for the target system (**cust.idl**).
3. Creating the e*Gate schema components or importing the schema from the CD-ROM.
4. Compiling the IDL into the CORBA client application.
5. Running e*Gate.

4.2.1 Creating the Target System

In the example, the target system called **CUST** is a simple application for holding customer details. All records are stored in a flat file using a carat "^" to separate the fields. The fields are a unique ID, first name, last name (or surname) and address.

To create the target system:

1. Create the following text file:

```
301 ^Alan ^Aspen ^23 Acacia Avenue, London, UK
430 ^Brian ^Browning ^7a Brewer St., Monrovia, California, USA
799 ^Carla ^Crowthorne ^87 Crown St., Melbourne, Australia
286 ^Debbie ^Davidson ^107 Derby Road, Cape Town, South Africa
137 ^John ^Browning ^23 Bell Lane, Bournemouth, UK
```

2. Save the file as **c:\Temp\cust.dat**.

Important: *Make sure the* ***cust.dat*** *file does not include any empty carriage returns.*

4.2.2 Writing the IDL and Starting the Interface Repository

The next step is to determine the type of query you want to make of the target system, the associated data types and the types of error conditions that could exist. This will enable you to write the corresponding IDL specification.

For the example, we will define two methods, both of them to read data from **CUST**. (Note that you could also implement methods that set data in the target system). The two methods belong to an interface representing the target system known as **CUST**, which exists within a module called **eGate**. Specifying the module is important because it enables the CORBA-VisiBroker Server e*Way to determine which interfaces to implement at runtime.

To create the IDL file:

1. Create the following text file:

```
/*
 * File cust.idl
 */
```

```
// The e*Way will emulate all interfaces in modules beginning with
"eGate" (the module qualifier)
module eGate {
    interface Cust {

        struct customer {
            short id;
            string firstname;
            string surname;
            string address;
        };

        typedef sequence<customer> customerList;

        exception custProblem {
            string reason;
        };

        string getAddress(in string firstname, in string surname)
raises(custProblem);

        customerList getCustomers(in string surname)
raises(custProblem);
    };
};
```

2 Save the file as **cust.idl**.

The IDL defines a struct customer (representing one row of the file) and has two methods. The first method, **getAddress**, returns an address for a customer with a given first and last name (surname). The second method, **getCustomers**, returns a sequence of all customers with the given last name.

To load the IDL into an Interface Repository:

- From a command line, type the following command:

```
irep -console EGATEREP cust.idl
```

Note: *If you choose a different name for the Interface Repository, substitute the name for **EGATEREP** appropriately.*

The next step in the process is to create the e*Gate schema. To create the schema components yourself, see **"Creating the Sample Schema"** below. To import the schema from the CD-ROM, see **"Importing the Sample Schema" on page 34**.

4.2.3 Creating the Sample Schema

To implement the CORBA-VisiBroker Server e*Way within an e*Gate system, launch the e*Gate Schema Designer and create a new schema called **CORBAServer**. Then, do the following:

- Create Event Types and generate the Event Type Definitions (ETDs) required for each of the methods.
- Create Collaboration Rules and Collaboration Rules scripts to process Event data.
- Define the IQs to which Event data will be published prior to sending it to the external system.

- Create a new e*Way component and configure its properties. Use the e*Way Editor to configure its parameters.
- Within the e*Way component, create and configure Collaborations to apply the required Collaboration Rules.

Creating Event Types and Event Type Definitions

The sample schema uses three Event Types: **msg**, **getCustomers**, and **getAddress**. **msg** carries the data to and from the external system. **getCustomers** and **getAddress** represent the CORBA methods.

To create the Event Type msg:

1. In the Navigator, select the **Components** tab if it is not already elected.
2. In the Navigator, select the **Event Types** folder.
3. Create a new Event Type called **msg**.
4. Display the properties for Event Type msg.
5. Under **Event Type Definition** click **New**.

   The ETD Editor will launch.
6. Create a new Event Type Definition, msg.ssc, with a single root node.
7. Save the file and promote it to run time.
8. Exit the ETD Editor and click **OK** in the Event Type Properties dialog box to return to the e*Gate Schema Designer's main screen.

To create the Event Types getCustomers and getAddress:

1. Create a new Event Type called **getCustomers**.
2. Display the properties for Event Type **getCustomers**.
3. Under **Event Type Definition** click **New**.

   The ETD Editor will launch.
4. In the **New** dialog box, type **getCustomers**. Do not specify any file extension—the Editor will supply the .ssc extension automatically.
5. Use the CORBA VisiBroker Server Converter Build tool to create the Event Type Definition **getCustomers.ssc**. For information on using the Build tool, see **"Using the ETD Editor's Build Tool" on page 19**.
6. Save the file and promote it to run time.
7. Exit the ETD Editor and click **OK** in the Event Type Properties dialog box to return to the e*Gate Schema Designer's main screen.
8. Repeat the above 7 steps for **getAddress**.

Create the Collaboration Rules and Collaboration Rules Scripts

In this example, the CUST system is represented by a flat file and a single e*Way is used to perform the query. The example can be extended to connect to other systems such as databases, using separate e*Ways, BOBs, extra Collaborations and IQs. To do this, you need to copy the 24-byte node called “Envelope” in all your Collaboration Rules scripts. This allows e*Gate to ensure that a reply received from the target system is matched with the correct CORBA client request.

The sample schema requires four Collaboration Rules and Collaboration Rules scripts. Two use the Monk ID Collaboration Service and two use the Monk Collaboration Service.

To create the Collaboration Rules gAddress:

1 In the Navigator, select the **Collaboration Rules** folder.

2 Create a new Collaboration Rules component called **gAddress**.

3 Edit the properties of **gAddress** as follows:

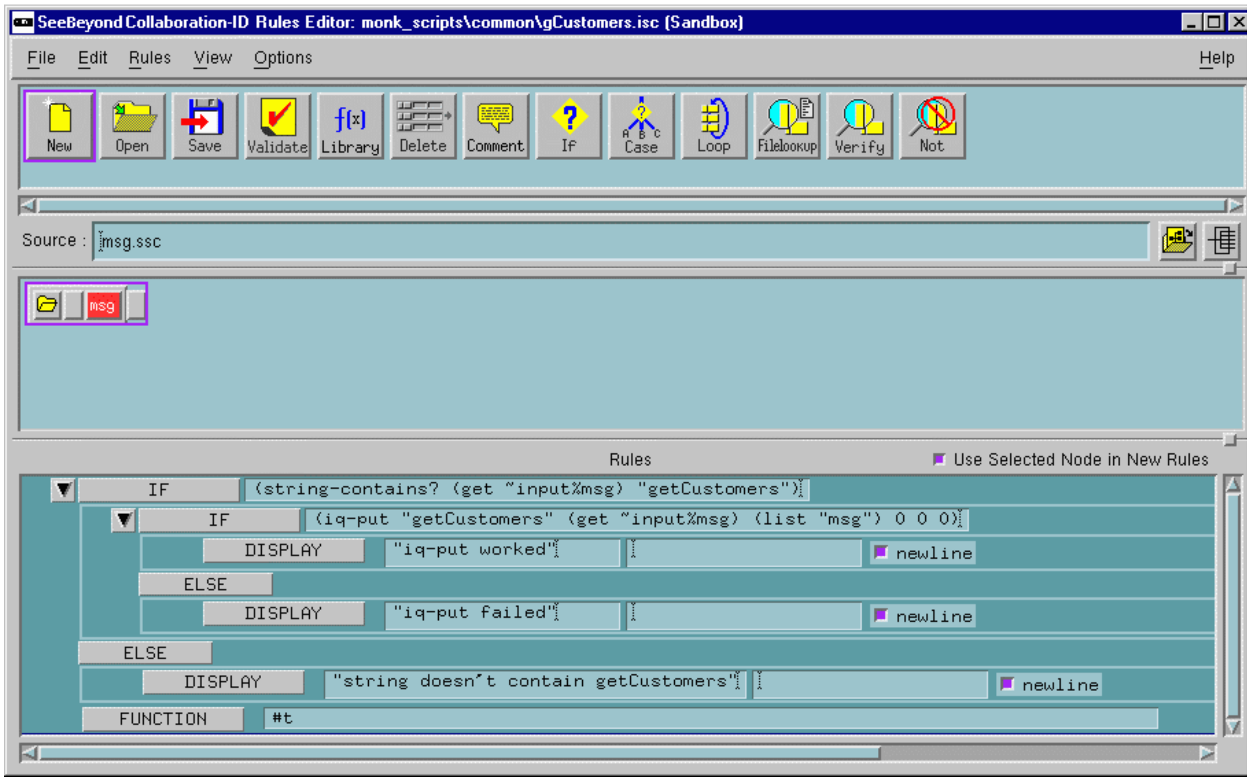
Service	Monk ID
Subscription	**msg**
Publication	**getAddress**

4 On the **General** tab, under **Collaboration Rules**, click **New**.

The SeeBeyond Collaboration-ID Rules Editor will launch.

5 Create a new file called **gAddress.isc**. The following screen shot illustrates the Rules for this file.

Figure 9 gAddress.isc


SeeBeyond Collaboration-ID Rules Editor: monk_scripts\common\gCustomers.isc (Sandbox)
File
Edit
Rules
View
Options
Help
New
Open
Save
Validate
Library
Delete
Comment
If
Case
Loop
Filelookup
Verify
Not
Source :
msg
Rules
Use Selected Node in New Rules

msg.ssc

```
IF        (string-contains? (get ~input%msg) "getCustomers")
    IF        (iq-put "getCustomers" (get ~input%msg) (list "msg") 0 0 0)
        DISPLAY   "iq-put worked"                 newline
    ELSE
        DISPLAY   "iq-put failed"                 newline
ELSE
    DISPLAY   "string doesn't contain getCustomers"        newline
FUNCTION  #t
```

6 Save the file and promote it to run time.

7 Exit the SeeBeyond Collaboration-ID Rules Editor and click **OK** in the Collaboration Rules Properties dialog box to return to the e*Gate Schema Designer's main screen.

To create the Collaboration Rules gCustomers:

1 Create a new Collaboration Rules component called **gCustomers**.

2 Edit the properties of **gCustomers** as follows:

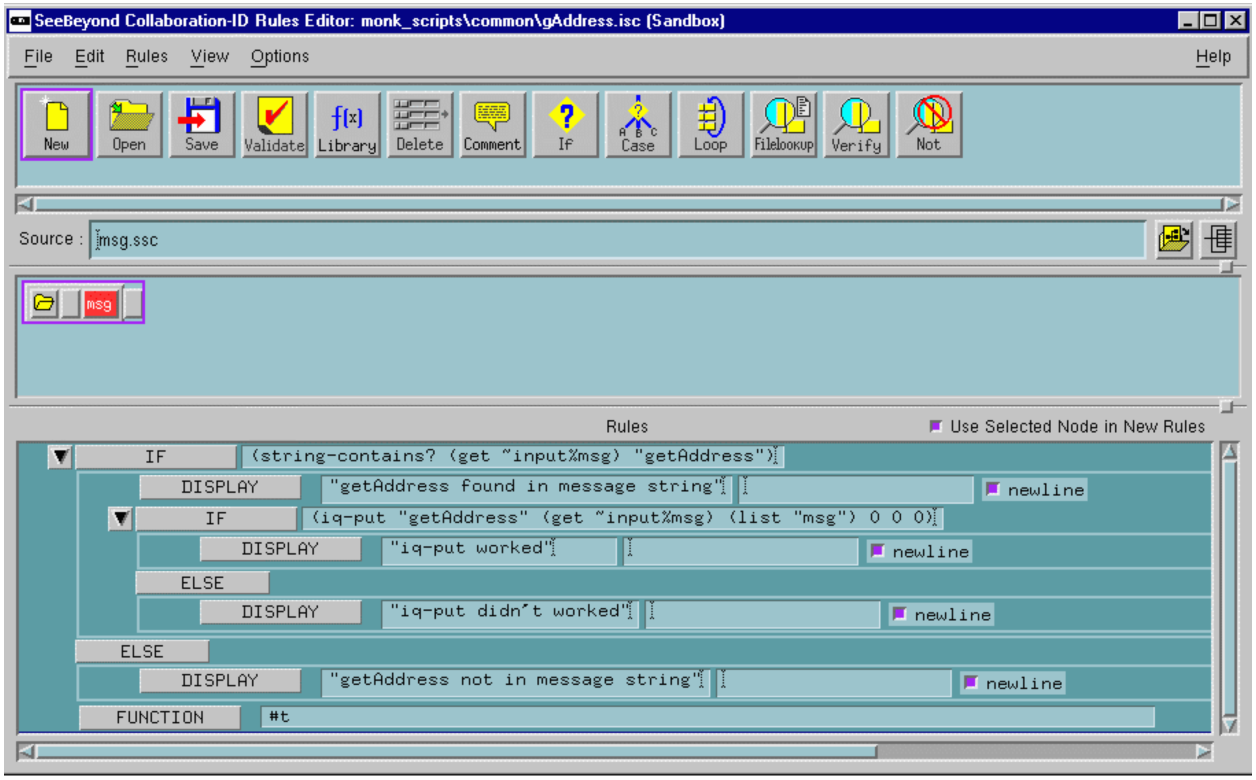
Service	Monk ID
Subscription	**msg**
Publication	**getCustomers**

3 On the **General** tab, under **Collaboration Rules**, click **New**.

The SeeBeyond Collaboration-ID Rules Editor will launch.

4 Create a new file called **gCustomers.isc**. The following screen shot illustrates the Rules for this file.

Figure 10 gCustomers.isc


SeeBeyond Collaboration-ID Rules Editor: monk_scripts\common\gAddress.isc (Sandbox)
File
Edit
Rules
View
Options
Help
New
Open
Save
Validate
Library
Delete
Comment
If
Case
Loop
Filelookup
Verify
Not
msg
Rules
Use Selected Node in New Rules

Source : msg.ssc

```
IF        (string-contains? (get ~input%msg) "getAddress")
  DISPLAY   "getAddress found in message string"          newline
  IF        (iq-put "getAddress" (get ~input%msg) (list "msg") 0 0 0)
    DISPLAY   "iq-put worked"                             newline
  ELSE
    DISPLAY   "iq-put didn't worked"                      newline
ELSE
  DISPLAY   "getAddress not in message string"            newline
  FUNCTION  #t
```

5 Save the file and promote it to run time.

6 Exit the SeeBeyond Collaboration-ID Rules Editor and click **OK** in the Collaboration Rules Properties dialog box to return to the e*Gate Schema Designer's main screen.

To create the Collaboration Rules getAddress:

1 In the Navigator, select the **Collaboration Rules** folder.

2 Create a new Collaboration Rules component called **getAddress**.

3 Edit the properties of **getAddress** as follows:

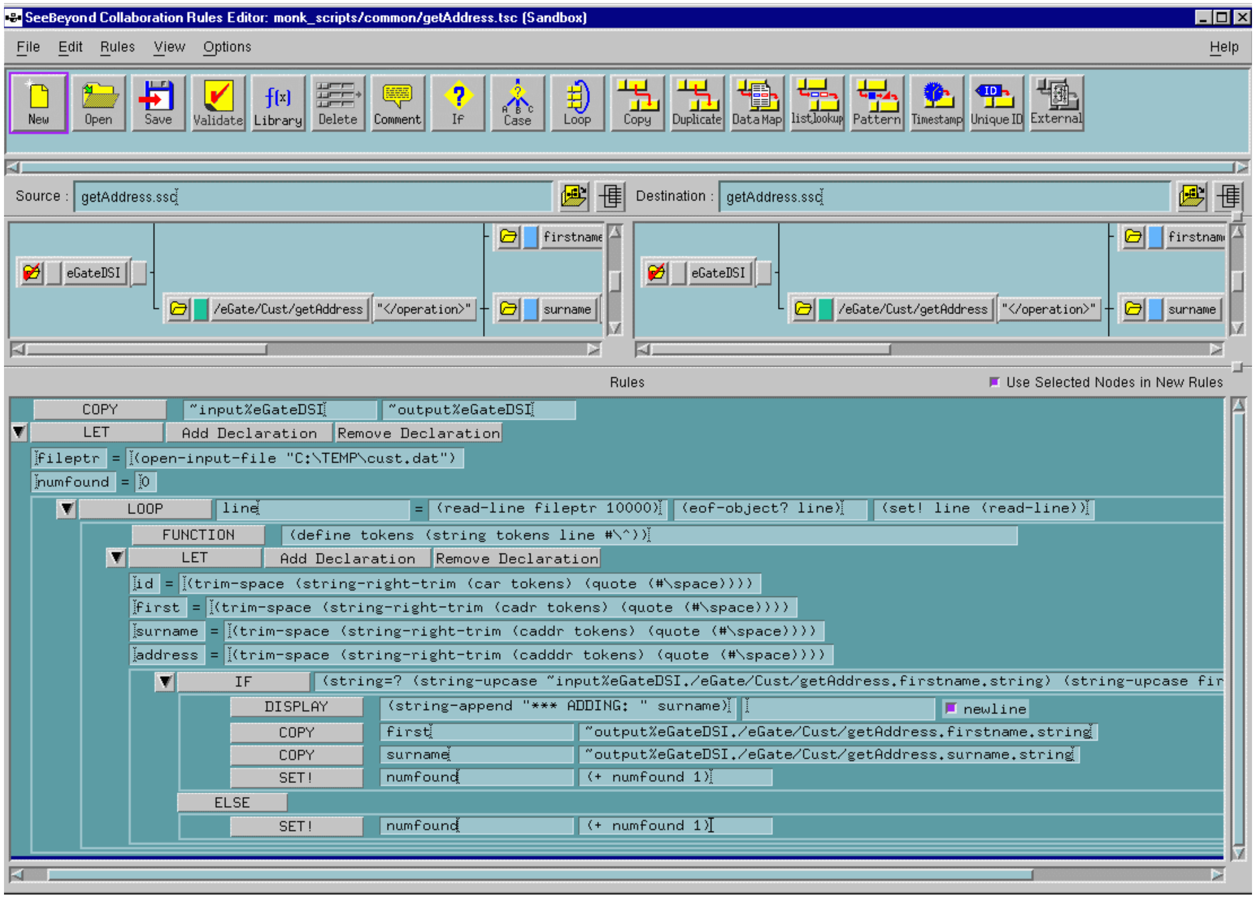
Service	Monk
Subscription	**getAddress**
Publication	**getAddress**

4 On the **General** tab, under **Collaboration Rules**, click **New**.

The SeeBeyond Collaboration Rules Editor will launch.

5 Create a new file called **getAddress.tsc**. The following screen shot illustrates the Rules for this file.

Figure 11 getAddress.tsc



Source : getAddress.ssc

Destination : getAddress.ssc

```
COPY    ~input%eGateDSI    ~output%eGateDSI
LET     Add Declaration    Remove Declaration
  fileptr = (open-input-file "C:\TEMP\cust.dat")
  numfound = 0
  LOOP    line = (read-line fileptr 10000)    (eof-object? line)    (set! line (read-line))
    FUNCTION    (define tokens (string tokens line #\^))
    LET     Add Declaration    Remove Declaration
      id = (trim-space (string-right-trim (car tokens) (quote (#\space))))
      first = (trim-space (string-right-trim (cadr tokens) (quote (#\space))))
      surname = (trim-space (string-right-trim (caddr tokens) (quote (#\space))))
      address = (trim-space (string-right-trim (cadddr tokens) (quote (#\space))))
      IF    (string=? (string-upcase ~input%eGateDSI./eGate/Cust/getAddress.firstname.string) (string-upcase fir
          DISPLAY   (string-append "*** ADDING: " surname)        newline
          COPY      first       ~output%eGateDSI./eGate/Cust/getAddress.firstname.string
          COPY      surname     ~output%eGateDSI./eGate/Cust/getAddress.surname.string
          SET!      numfound    (+ numfound 1)
      ELSE
          SET!      numfound    (+ numfound 1)
```

6 Save the file and promote it to run time.

7 Exit the SeeBeyond Collaboration Rules Editor and click **OK** in the Collaboration Rules Properties dialog box to return to the e*Gate Schema Designer's main screen.

To create the Collaboration Rules getCustomers:

1 Create a new Collaboration Rules component called **getCustomers**.

2 Edit the properties of **getCustomers** as follows:

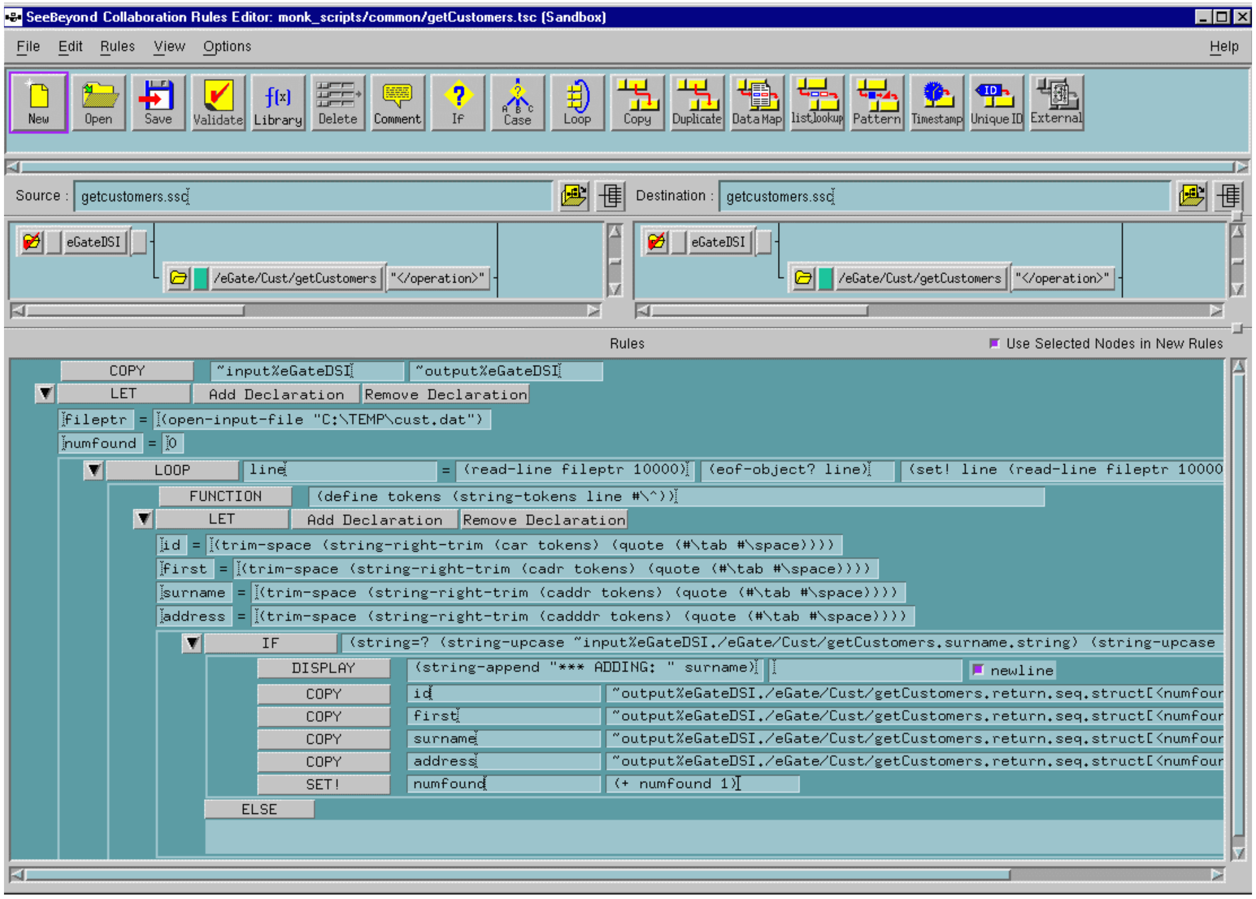
Service	Monk
Subscription	**getCustomers**
Publication	**getCustomers**

3 On the **General** tab, under **Collaboration Rules**, click **New**.

The SeeBeyond Collaboration-ID Rules Editor will launch.

4 Create a new file called **getCustomers.tsc**. The following two screen shots illustrate the Rules for this file.

Figure 12 getCustomers.tsc – Part 1

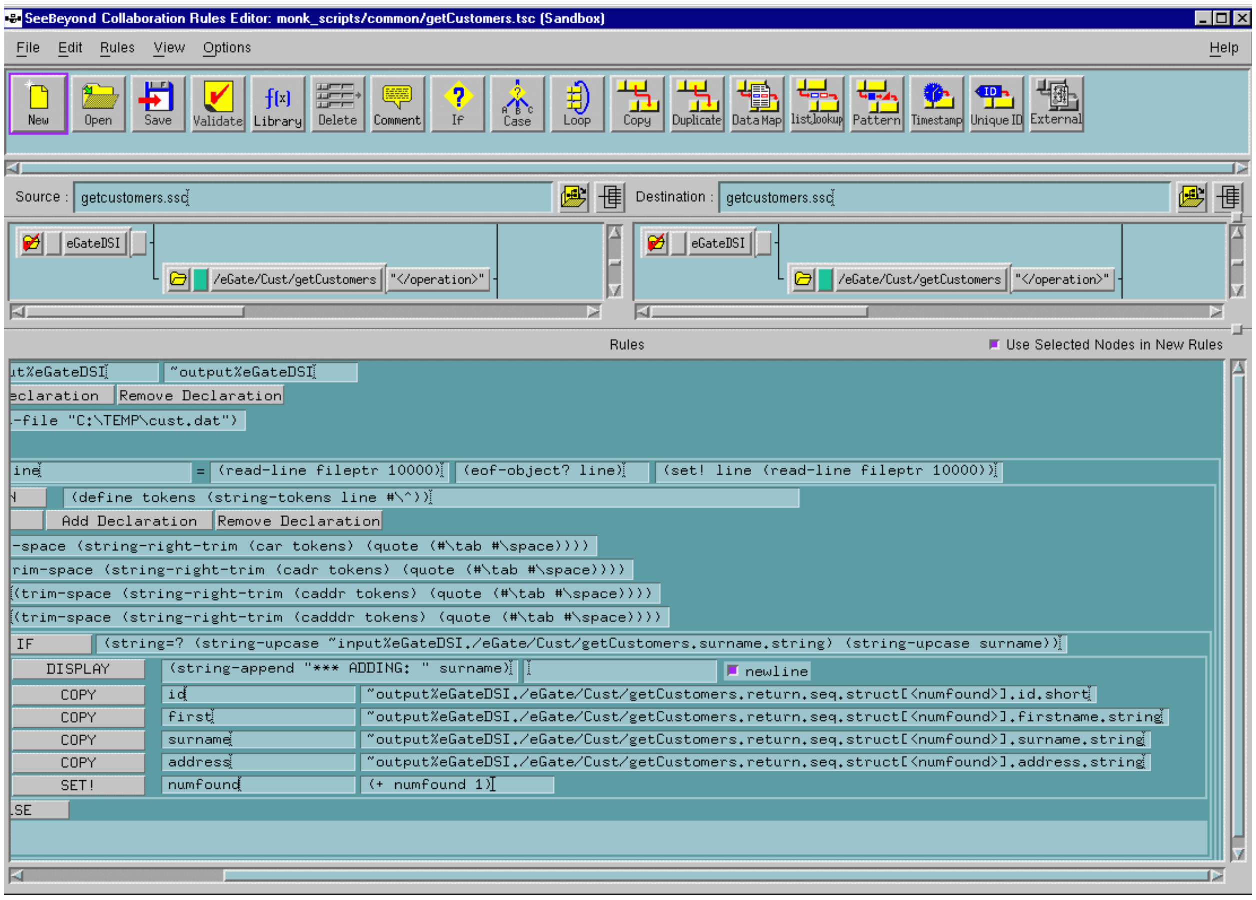

SeeBeyond Collaboration Rules Editor: monk_scripts/common/getCustomers.tsc (Sandbox)
File Edit Rules View Options Help
New Open Save Validate Library Delete Comment If Case Loop Copy Duplicate Data Map list lookup Pattern Timestamp Unique ID External
eGateDSI
eGateDSI
Rules
Use Selected Nodes in New Rules

Source : getcustomers.ssc
Destination : getcustomers.ssc

/eGate/Cust/getCustomers "</operation>"
/eGate/Cust/getCustomers "</operation>"

```
COPY   ~input%eGateDSI   ~output%eGateDSI
LET   Add Declaration   Remove Declaration
  fileptr = (open-input-file "C:\TEMP\cust.dat")
  numfound = 0
    LOOP   line = (read-line fileptr 10000)   (eof-object? line)   (set! line (read-line fileptr 10000
      FUNCTION   (define tokens (string-tokens line #\^))
      LET   Add Declaration   Remove Declaration
        id = (trim-space (string-right-trim (car tokens) (quote (#\tab #\space))))
        first = (trim-space (string-right-trim (cadr tokens) (quote (#\tab #\space))))
        surname = (trim-space (string-right-trim (caddr tokens) (quote (#\tab #\space))))
        address = (trim-space (string-right-trim (cadddr tokens) (quote (#\tab #\space))))
          IF   (string=? (string-upcase ~input%eGateDSI./eGate/Cust/getCustomers.surname.string) (string-upcase
            DISPLAY   (string-append "*** ADDING: " surname)      newline
            COPY   id        ~output%eGateDSI./eGate/Cust/getCustomers.return.seq.struct[<numfour
            COPY   first     ~output%eGateDSI./eGate/Cust/getCustomers.return.seq.struct[<numfour
            COPY   surname   ~output%eGateDSI./eGate/Cust/getCustomers.return.seq.struct[<numfour
            COPY   address   ~output%eGateDSI./eGate/Cust/getCustomers.return.seq.struct[<numfour
            SET!   numfound  (+ numfound 1)
          ELSE
```

Figure 13 getCustomers.tsc – Part 2



Source : getcustomers.ssc
Destination : getcustomers.ssc

```
eGateDSI
  /eGate/Cust/getCustomers  "</operation>"
```

```
ut%eGateDSI    ~output%eGateDSI
eclaration    Remove Declaration
-file "C:\TEMP\cust.dat")

ine    =  (read-line fileptr 10000)  (eof-object? line)  (set! line (read-line fileptr 10000))
N    (define tokens (string-tokens line #\^))
     Add Declaration  Remove Declaration
-space (string-right-trim (car tokens) (quote (#\tab #\space))))
rim-space (string-right-trim (cadr tokens) (quote (#\tab #\space))))
(trim-space (string-right-trim (caddr tokens) (quote (#\tab #\space))))
(trim-space (string-right-trim (cadddr tokens) (quote (#\tab #\space))))
IF        (string=? (string-upcase ~input%eGateDSI./eGate/Cust/getCustomers.surname.string) (string-upcase surname))
 DISPLAY  (string-append "*** ADDING: " surname)    newline
 COPY     id        ~output%eGateDSI./eGate/Cust/getCustomers.return.seq.struct[<numfound>].id.short
 COPY     first     ~output%eGateDSI./eGate/Cust/getCustomers.return.seq.struct[<numfound>].firstname.string
 COPY     surname   ~output%eGateDSI./eGate/Cust/getCustomers.return.seq.struct[<numfound>].surname.string
 COPY     address   ~output%eGateDSI./eGate/Cust/getCustomers.return.seq.struct[<numfound>].address.string
 SET!     numfound  (+ numfound 1)
SE
```

Note: *Each record found is added to a sequence. Therefore, you need to use the variable "numbound" to index the four fields that are being copied (ID, first name, surname and address).*

5 Save the file and promote it to run time.

6 Exit the SeeBeyond Collaboration Rules Editor and click **OK** in the Collaboration Rules Properties dialog box to return to the e*Gate Schema Designer's main screen.

Defining IQs

1 In the Navigator, double-click the **Participating Hosts** folder, then double-click the host, then double-click the Control Broker and select the IQ Manager.

2 Display the properties for the IQ Manager and configure it to start automatically.

3 Create an IQ called **iq**.

4 Display the IQ's properties and verify that it uses the SeeBeyond Standard IQ Service.

Defining and Configuring the e*Way

The sample schema uses a single CORBA-VisiBroker Server e*Way to receive data from the CORBA client and send it to the target system.

To define the e*Way:

1. In the Navigator, double-click the **Participating Hosts** folder, then double-click the host and select the Control Broker.
2. Create a new e*Way component called **Server**.
3. Display the e*Way's properties.
4. On the **General** tab, under **Executable file**, click **Find** to assign the file **stcewfwmux.exe**.
5. Select the **Start Up** tab. Configure the e*Way to start automatically.

To configure the e*Way's parameters:

1. Return to the **General** tab. Under **Configuration file**, click **New** to launch the e*Way Editor.
2. Configure the e*Way's parameters in the e*Way Editor as follows (parameters not listed in the table should retain their default values):

Section	Parameter: Setting
Server Settings	Server Name: The name of the Participating Host (for example, **localhost**)
	Server Interface Repository Name: **EGATEREP**
	Auto Start Server: **Yes**
	Server Exec String: **vbj com.stc.dsi.EgateServer -ir EGATEREP -modq eGate**
	Classpath: The value of the CLASSPATH variable. For more information see **"Setting the CLASSPATH" on page 17**.

3. Save the settings and promote the file to run time.
4. In the e*Way's Properties dialog box, click **OK** to save all changes and return to the e*Gate Schema Designer's main window.

Defining Collaborations

The **Server** e*Way contains four Collaborations (two for each CORBA method) to process the call from the CORBA client. You must configure the Collaborations in the order listed in order for the Event Types to be published before the next component subscribes to them.

The order in which you should create and configure the Collaborations is:

- **gAddress**
- **getAddress**

- **gCustomers**
- **getCustomers**

To create the Collaboration gAddress:

1. In the Navigator, select the e*Way **Server**.
2. Create a Collaboration called **gAddress**.
3. Display the Collaboration's properties and edit them as follows:

Collaboration Rules	**gAddress**
Subscription	Event Type: **msg** Source: **<EXTERNAL>**
Publication	Event Type: **getAddress** Destination **iq**

4. In the Collaboration's Properties dialog box, click **OK** to save all changes and return to the e*Gate Schema Designer's main window.

To create the Collaboration getAddress:

1. In the Navigator, select the e*Way **Server**.
2. Create a Collaboration called **getAddress**.
3. Display the Collaboration's properties and edit them as follows:

Collaboration Rules	**getAddress**
Subscription	Event Type: **getAddress** Source: **gAddress**
Publication	Event Type: **getAddress** Destination **<EXTERNAL>**

4. In the Collaboration's Properties dialog box, click **OK** to save all changes and return to the e*Gate Schema Designer's main window.

To create the Collaboration gCustomers:

1. In the Navigator, select the e*Way **Server**.
2. Create a Collaboration called **gCustomers**.
3. Display the Collaboration's properties and edit them as follows:

Collaboration Rules	**gCustomers**
Subscription	Event Type: **msg** Source: **<EXTERNAL>**
Publication	Event Type: **getCustomers** Destination **iq**

4. In the Collaboration's Properties dialog box, click **OK** to save all changes and return to the e*Gate Schema Designer's main window.

To create the Collaboration getAddress:

1. In the Navigator, select the e*Way **Server**.
2. Create a Collaboration called **getCustomers**.
3. Display the Collaboration's properties and edit them as follows:

Collaboration Rules	**getCustomers**
Subscription	Event Type: **getCustomers** Source: **gCustomers**
Publication	Event Type: **getCustomers** Destination **<EXTERNAL>**

4. In the Collaboration's Properties dialog box, click **OK** to save all changes and return to the e*Gate Schema Designer's main window.

This completes schema configuration in the e*Gate Schema Designer.

4.2.4 Importing the Sample Schema

Instead of creating and configuring all of the schema components yourself, you can import the sample schema directly from the CD-ROM.

Importing the Sample Schema.

To import the sample schema:

1. Start e*Gate Schema Designer and create a new schema.
2. On the **File** menu, click **Import Definitions from File**.
3. On the **Import Wizard Introduction** window, click **Next**.
4. Click **Schema**, then click **Next**.
5. Specify the sample schema file name, and then click **Next**. The installation CD path name of this file should be **\samples\ewvbcorbaserver\ewvbcorbaserver.zip**.

Configuring the e*Way

Once you import the schema you need to configure the e*Way component before running data through e*Gate.

To configure the e*Way:

1. In the Navigator, select the **Components** tab if it is not already selected.
2. In the Navigator, select the **Participating Hosts** folder and navigate to the e*Way components.
3. Select the **Server** e*Way and display its properties.
4. Under **Configuration file**, click **Edit** to launch the e*Way Editor.

5 Configure the e*Way's parameters in the e*Way Editor as follows (parameters not listed in the table should retain their default values):

Section	Parameter: Setting
Server Settings	Server Name: The name of the Participating Host (for example, **localhost**)
	Server Interface Repository Name: **EGATEREP**
	Auto Start Server: **Yes**
	Server Exec String: **vbj com.stc.dsi.EgateServer -ir EGATEREP -modq eGate**
	Classpath: The value of the CLASSPATH variable. For more information see **"Setting the CLASSPATH" on page 17**.

6 Save the settings and promote the file to run time.

7 In the e*Way's Properties dialog box, click **OK** to save all changes and return to the e*Gate Schema Designer's main window.

4.2.5 Compiling the IDL Into the CORBA Client Application

Listed below is the test application in full. It is written in Java but any supported language such as C++ would be suitable.

To extract the file from the CD-ROM, follow the **procedure on page 34**. The name of the file is **EgateClient.java**.

EgateClient.java

```
import org.omg.*;


public class EgateClient {

    static org.omg.CORBA.ORB orb = null;

    static eGate.Cust cust = null;


public static void getCustomers(String surname) {
    try {

System.out.println("\n*******************************************");
        System.out.println("Calling cust.getCustomers(" + surname +
")...");
        eGate.CustPackage.customer[] customers =
cust.getCustomers(surname);
        int numcusts = customers.length;
        System.out.println("**** Found " + numcusts + " customers:-
");
```

```
            for (int i = 0; i < numcusts; i++) {
                System.out.println("* Customer " + (i + 1));
                System.out.println("     id          = " +
    customers[i].id);
                System.out.println("     first name = " +
    customers[i].firstname);
                System.out.println("     surname    = " +
    customers[i].surname);
                System.out.println("     address    = " +
    customers[i].address);
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
    public static void main(String[] args) {
        orb = org.omg.CORBA.ORB.init(args, null);
        System.out.println("");
        System.out.println("-----------------------------");
        System.out.println("Binding to eGate.CustHelper");
        cust = eGate.CustHelper.bind(orb);
        System.out.println("-----------------------------");
        try {
            getCustomers("Aspen");
            getCustomers("Browning");
            getCustomers("Clark");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
    }
```

To compile the IDL:

- From a command line type the following:

```
idl2java cust.idl
```

To compile the client application:

- From a command line type the following:

```
javac -deprecation EgateClient.java
```

4.2.6 Running the Sample

Running the sample involves two separate tasks: Running e*Gate and invoking the client application.

Note: *Before running the sample, make sure the target system is available and the Interface Repository is running. See* **"Creating the Target System" on page 24** *and* **"Writing the IDL and Starting the Interface Repository" on page 24** *for more information about these tasks.*

To run e*Gate:

- Start the Control Broker for the **CORBAServer** schema. For information on starting and monitoring schema components, see the *e*Gate Integrator System Administration and Operations Guide.*

To invoke the client application:

- Ensure that the VisiBroker java files are in your CLASSPATH variable. Then, type the following on a command line:

  ```
  vbj EgateClient
  ```

  You should see the following output:

```
-----------------------------
Binding to eGate.CustHelper
-----------------------------

*******************************************
Calling cust.getCustomers(Aspen)...
**** Found 1 customers:-
* Customer 1
     id         = 301
     first name = Alan
     surname    = Aspen
     address    = 23 Acacia Avenue, London, UK

*******************************************
Calling cust.getCustomers(Browning)...
**** Found 2 customers:-
* Customer 1
     id         = 430
     first name = Brian
     surname    = Browning
     address    = 7a Brewer St., Monrovia, USA
* Customer 2
     id         = 137
     first name = John
     surname    = Browning
     address    = 23 Bell Lane, Bournemouth, UK

*******************************************
Calling cust.getCustomers(Clark)...
**** Found 0 customers:-
```

4.3 Starting the CORBA-VisiBroker Server e*Way

To start the CORBA-VisiBroker Server e*Way, both the stcewfwmux component process and the CORBA Server component process must be started.

To autostart the stcewfwmux component process:

1. From the e*Way Properties dialog, select the **Start Up** tab.
2. Select the **Start automatically** check box to start the stcewfwmux e*Way component when the Control Broker is started.

To autostart the CORBA Server component process after the stcewfwmux component process is started refer to the parameter **"Auto Start Server" on page 15**. If the autostart option is not selected at this time, you must start the component manually before invoking the client application.

To manually start the CORBA Server component process:

1. Start a command-line session.

2 Type the following command:

```
vbj com.stc.dsi.EgateServer
```

4.4 CORBA-VisiBroker Supported Types

The CORBA-VisiBroker Server e*Way supports the following IDL types as IN, OUT and INOUT arguments to methods, return values and within user-defined exceptions:

- Boolean
- Char
- Double
- Float
- Long
- Long long
- Sequence
- Short
- Struct *(Note: for* **Struct**, OUT *is not supported)*
- Void (as return type)

Note: *One-way method invocations are not supported in this release—all CORBA invocations are request-reply.*

Chapter 5

Troubleshooting

This chapter discusses troubleshooting for the CORBA-VisiBroker Server e*Way.

5.1 General Troubleshooting

5.1.1 Failure to Connect to External Application

Problem

The following exception is generated within the CORBA application:

```
"org.omg.CORBA.NO_IMPLEMENT[completed=MAYBE]"
```

Solution

The exception occurs when your client application cannot find the object it needs. Make sure that the configuration for the e*Way specifies that it should AutoStart the CORBA Server component. If you chose to start the CORBA Server component by hand, make sure that it is running.

Check to ensure the interface repository ('irep') is running and that it contains the correct IDL. Check the log files to ensure that the CORBA Server component read the interface repository correctly on start-up.

5.2 CORBA Application Hangs When Making a Call

Problem

The client application binds to the CORBA object and then hangs, waiting for it to return.

Solution

This problem occurs when e*Gate does not receive a reply from the target system. Check the log files to see if a Collaboration failed for the particular query or to see if the target system is down.

Index