SeeBeyond ICAN Suite

eGate Integrator User's Guide

Release 5.0.5

Version 20041118184210.

Contents

Figures

Tables

Chapter 1

Introduction

This chapter describes the new features in this release of eGate Integrator, the scope and organization of this document, and provides references to additional sources of relevant information.

What's in This Chapter

1.1 About eGate Integrator

SeeBeyond® eGate™ Integrator is an advanced, distributed integration platform that serves as the foundation of the SeeBeyond Integrated Composite Application Network™ (ICAN™) Suite. The user interfaces—Enterprise Designer for Project design and Enterprise Manager for browser-based installation, management, and monitoring— ensure a unified look and feel across all editors in the suite, with a single sign-on process for access to any product.

During Project design, projects are made more maintainable by separating views of logical components from views of the physical environment—the Project Explorer view and the Environment Explorer view.

The run-time environment, which is J2EE-compatible and certified, features high performance and dynamic scalability. The architecture uses EJB (Enterprise Java Beans) with JMS (Java Message Service) and JNDI (Java Naming and Directory Interface). Using JCA (Java Connector Architecture), configurations are deployable to J2EE-compliant implementations, such as WebLogic 8.1 and WebSphere 5.

1.2 What's New in This Release

For information regarding operating system support, language support, and performance improvements, please refer to the *SeeBeyond ICAN Suite Installation Guide* and *eGate Integrator Release Notes*.

1.2.1 LDAP Support

- eGate Integrator now supports OpenLDAP Directory Server as well as Microsoft's Active Directory and Sun Microsystem's Sun Java System Directory Server. In addition, LDAPS (LDAP over SSL) is supported for encrypted communications between the Repository and all three supported directory servers.
- To support connections to LDAP servers that store information in a hierarchical structure, new sub-tree search properties were added to the LDAP server Properties sections of the Integration Server Properties and JMS IQ Manager Properties dialogs.

1.2.2 Enterprise Designer

Basic User Interface

- In Project Explorer and applicable editors, the default *SeeBeyond* folder now has a unique icon to distinguish it from all other projects.
- In Environment Explorer, the context menu items under *New…* (such as the list of external systems) are now sorted alphabetically.
- The extended language option now supports additional character sets for Japanese, Korean, and Chinese locales.

Collaboration Definition Editor (Java)

- The Business Rules Designer toolbar has a revised layout with additional tools, and objects on the canvas of the Business Rules Designer now have tooltips.
- A new Class Browser is available for selecting classes, methods, and fields.
- New OTDs can now be added to an existing Collaboration by right-clicking the Collaboration in Project Explorer to display the Collaboration Definition Properties dialog.
- The properties of an existing method or variable in a Collaboration can now be edited, by right-clicking the method or variable and modifying the properties in the resulting dialog box.
- The new *Find* feature provides the ability to search for text strings within OTDs and variables in the scope of a selected business rule.
- New rules can be added to the start of a block (such as a **then** or **else** stanza) by selecting the parent node.

- Dragging and dropping tree nodes and method arguments within the Business Rules Designer now automatically creates canvas objects at the drop location.
- When input and output values are directly mapped in the Business Rules Designer and the destination node is repeating or optional, the repeating/optional element in which the field is set is now specified by an undefined index. This allows you to set a field in a specific element by defining the index.
- When you map between OTD nodes having different data types, if a conversion method exists for that pair of data types, a dialog box is automatically displayed in which you can specify certain properties associated with the conversion.
- A redesigned Collaboration Definition Tester offers new features:
  - Testing of both marshalable and unmarshalable OTDs is now supported.
  - Editing capabilities have been improved for optional and repeating nodes.
  - Java collaborations can be debugged within the tester when it is used in combination with the Java Source Editor. This allows line-by-line debugging of the Java code.

OTD Editor

- Additional Java methods for marshaling and unmarshaling have been added to the basic marshal() and unmarshal() methods for User-Defined OTDs; these are:
  - marshalToBytes()
  - marshalToString()
  - unmarshalFromBytes()
  - unmarshalFromString()

  The marshaling and unmarshaling processes differ from each other depending upon which Java method you use, and whether you are marshaling to or unmarshaling from byte[] or string fields.

  To support these methods, new data coding properties have been introduced:
  - Antecoding
  - Decoding
  - Encoding
  - Postcoding

  See **Specifying Data Coding** on page 161 for information regarding these new methods and coding properties..
- The new *maxOccurs* parameter for repeating nodes in User-Defined OTDs places a limit on the number of repetitions allowed for each node.
- The *Save As* command now allows saving an OTD to a different project.

Project Deployment and Activation

- In the Deployment Editor, the new *Automap* feature automatically assigns components to their matching containers when there is a one-to-one correspondence between them. This feature only works with external systems for

which it is enabled. To find out if Automap is enabled for a specific component, see the user documentation for that component.

- A warning prompt for deactivation is now issued before deleting a Project that has one or more active deployment profiles. (All active deployment profiles must be deactivated before the project can be deleted.)

Version Control

- The *Undo Checkout*, *Make Latest*, and *Version History: Retrieve Previous Version* options are now available for all eGate components, except WSDL OTDs.

Web Services

- Web Services and Web Services Clients now can support the Secure Sockets Layer (SSL). New properties have been added to the following configuration dialogs:
  - Integration Server Properties/Default Web Server
  - WebService External System

1.3 About This User's Guide

1.3.1 What's in This User's Guide

- **Chapter 1 "Introduction"** describes the purpose of this User's Guide, including writing conventions and a list of related documents.
- **Chapter 2 "System Overview"** provides an overview of the general structure, architecture, and operation of eGate Integrator, and it's place within the SeeBeyond ICAN Suite.
- **Chapter 3 "Enterprise Manager"** provides a detailed overview of the Enterprise Manager, including its structure and operation.
- **Chapter 4 "Enterprise Designer"** provides a detailed overview of the Enterprise Designer, including its structure and operation.
- **Chapter 5 "ICAN Projects"** explains how to create a Connectivity Map and use the Configuration Editor to modify eWay and JMS connections between Connectivity Map components.
- **Chapter 6 "Object Type Definitions"** describes how to create Object Type Definitions (OTDs).
- **Chapter 7 "Collaboration Definitions (Java)"** describes how to build Collaboration Definitions described in Java.
- **Chapter 8 "Collaboration Definitions (XSLT)"** describes how to build Collaboration Definitions described in XSLT.
- **Chapter 9"Environments"**explains how to create and populate eGate Integrator Environments, and how to configure and start Logical Hosts.
- **Chapter 10 "Project Deployment"** explains how to create and activate Deployment Profiles.
- **Chapter 11 "Web Services"** describes how to use eGate Integrator in concert with other ICAN Suite components to create Web Services.
- **Chapter 12 "Troubleshooting"** contains tips on troubleshooting eGate Integrator.

In addition, the **Glossary** on page 447 lists various terms used in this User's Guide.

1.3.2 Scope

This User's Guide provides general information about the features and operation of eGate Integrator in creating and deploying ICAN Projects. For information on eGate Integrator system management, see the *eGate Integrator System Administration Guide*.

Note: *Any operational explanations provided in this document are generic, for reference purposes only, and do not necessarily address the specifics of setting up individual ICAN Projects.*

1.3.3 Intended Audience

This User's Guide is intended for personnel who are involved in integrating software applications using eGate Integrator. To a large extent, these are individuals who will be using the eGate Integrator Enterprise Designer to build ICAN Projects to accomplish this task. This guide also provides a basic overview of the eGate Integrator product for those attempting to gain a general understanding of how eGate Integrator works.

This guide assumes that the reader is an experienced computer user, familiar with Windows-style GUI operations, and also has an in-depth understanding of the operating system(s) on which eGate Integrator will be installed.

1.3.4 Document Conventions

The following conventions are observed throughout this document.

Table 1 Writing Conventions

Text	Convention	Example
Button, file, icon, parameter, variable, method, menu, and object names.	**Bold** text.	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments and code samples	`Fixed` font. Variables are shown in ***`bold italic`***.	`bootstrap -p` ***`password`***
Hypertext links	**Blue** text.	**http://www.seebeyond.com**

1.3.5. Screenshots

Depending on what products you have installed, and how they are configured, the screenshots in this document may differ from what you see on your system.

1.4 Related Documents

The following SeeBeyond documents provide additional information about the eGate Integrator system as explained in this guide:

- *eGate Integrator JMS Reference Guide*
- *eGate Integrator Release Notes*
- *eGate Integrator System Administration Guide*
- *eGate Integrator Tutorial*

For information on a specific add-on product (for example, an eWay Intelligent Adapter), see the User's Guide for that product. A complete list of SeeBeyond documentation is included in the *SeeBeyond ICAN Suite Primer*.

The documentation for the SeeBeyond ICAN Suite is distributed as a collection of online documents, which can be accessed through the Enterprise Manager (see **Documentation** on page 46). These documents are in Adobe Acrobat format, which requires that Acrobat Reader be installed on your computer. Acrobat Reader can be from Adobe Systems as a free download from the following URL:

http://www.adobe.com

1.5 The 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

1.6 SeeBeyond Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

docfeedback@seebeyond.com

Chapter 2

System Overview

This chapter provides an overview of the conceptual operation and general architecture of the eGate Integrator system.

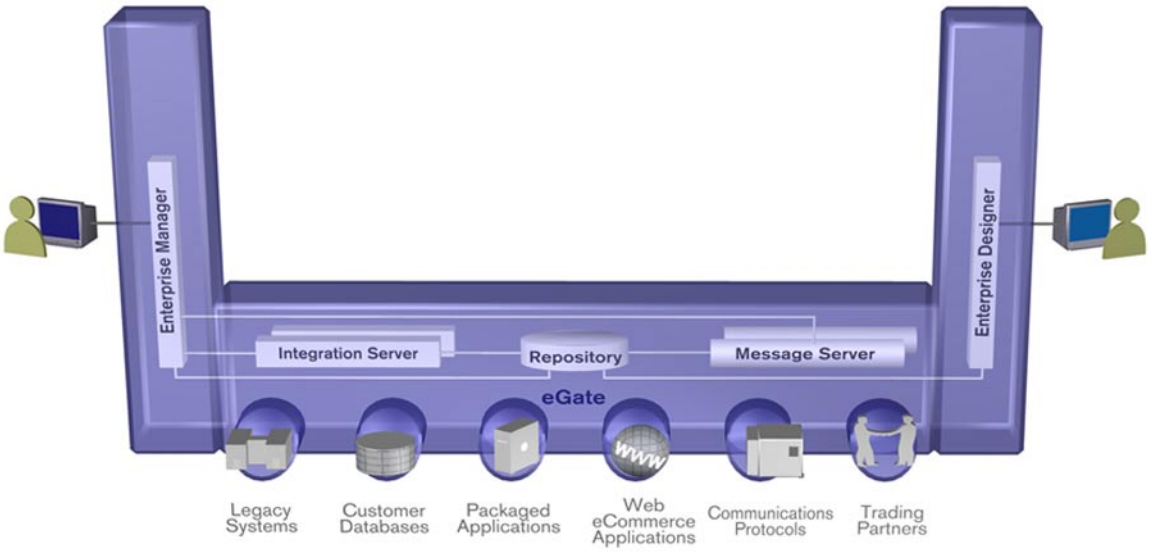
What's in This Chapter

2.1 Introduction

SeeBeyond® eGate™ Integrator is an advanced, distributed integration platform that serves as the foundation of the SeeBeyond Integrated Composite Application Network™ (ICAN™) Suite. It provides the core integration function, incorporating comprehensive systems connectivity, guaranteed messaging, and robust transformation capabilities. eGate Integrator also provides a unified, single sign-on environment for integration development, deployment, monitoring and management. eGate Integrator supports portability of integrations across common J2EE application servers through a completely open, J2EE-certified and Web Services-based architecture.

As shown in Figure 1, the heart of eGate Integrator is the Repository, which is a comprehensive store of information common to the entire enterprise. An integrated UDDI registry allows publication and discovery of Web Services. The run-time environment employs J2EE-compatible integration servers as operational engines and JMS-compatible message servers for the propagation of messages. The flexibility of the eGate Integrator system allows the option of deployment to a SeeBeyond run-time environment or to third-party application servers, across a distributed network of hardware platforms.

Figure 1 eGate Integrator



Enterprise Manager provides a unified, browser-based framework for managing all aspects of the run-time environment, as well as installing and updating all ICAN Suite components. Enterprise Designer provides a unified, graphical development environment for integrating systems and developing composite applications using Web Services.

eGate Integrator can communicate with and link multiple applications and databases across a variety of different operating systems. eGate Integrator performs with a wide variety of hardware, message standards, operating systems, databases, and communication protocols in both real-time and batch (scheduled) integration modes.

2.2 Integration Model

SeeBeyond addresses application integration by means of an ICAN Project, which contains the business logic required to solve the specific problem. The Project contains the various logical components and supporting information required to perform the routing, processing, and caching of messages containing the relevant data from one application to another. All Project information is stored in the Repository.

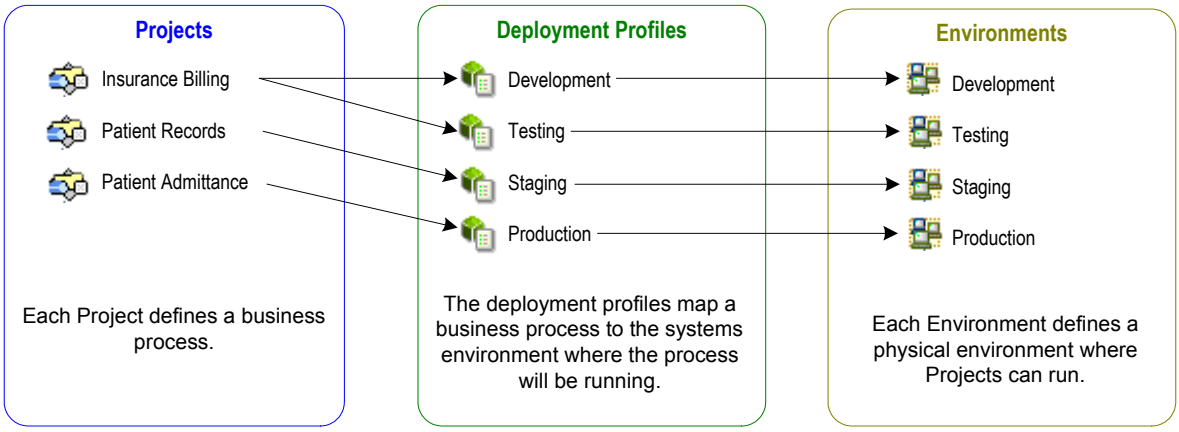
Projects are created using tools contained within Enterprise Designer and, once deployed, can be run and monitored using Enterprise Manager. Projects can also be set up to be run from the business process level using the SeeBeyond eInsight Business Process Manager, if that product is also installed.

Projects are run within individual sets of system definitions, referred to as Logical Hosts. These are defined within Environments, which represent the physical resources required to implement the Project. Projects are mapped to the individual Environments by means of deployment profiles, which are defined within the Enterprise Designer and become part of the Project. Activating the deployment profile deploys the Project to the associated Environment.

This structure of Projects, Environments, and deployment profiles isolates each implementation into logical and physical realms. This provides you with extensive flexibility and efficiency in designing eGate Integrator implementations. For example, once you build your Projects and Environments, you have the flexibility to change each realm without having to make changes to the other realm.

The finished Project, of course, will run in your production Environment; separate Environments, having the same structure as the production Environment, should be created for development and testing. You may also want some additional Environments, such as staging. The following figure illustrates the eGate Integrator implementation model using a healthcare-related example.

Figure 2 eGate Integrator Implementation Model



In the figure above, any of the Projects can be deployed to any of the Environments via the mapping defined in the deployment profiles. The example in the figure above shows that the patient admittance Project is already in the production phase and therefore was deployed using the production deployment profile. The patient records Project is in the staging phase and was therefore deployed to the staging Environment using the staging deployment profile. The insurance billing Project is still being developed and tested, and therefore it is deployed to development and testing via the development and testing profiles.

In broad outline, implementing an integration Project using eGate Integrator includes the following steps:

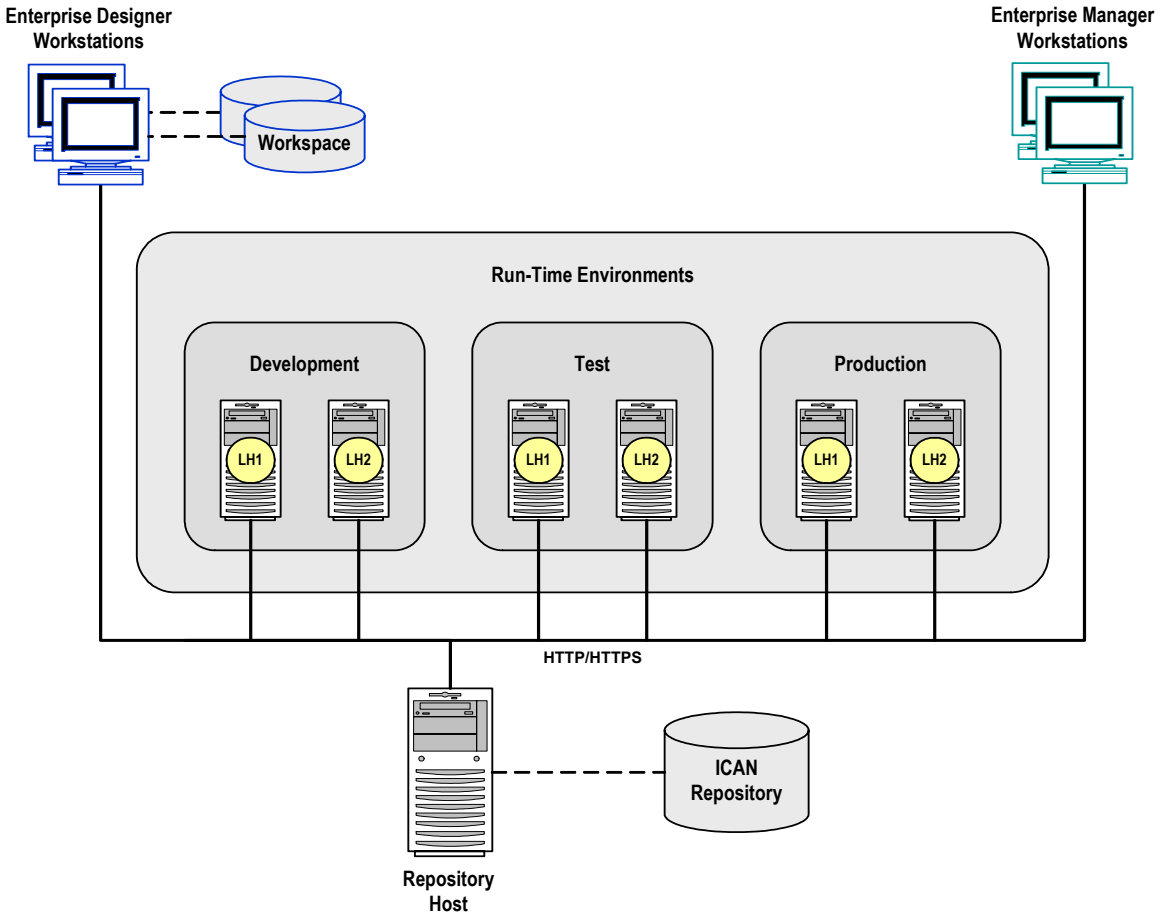
1. **Design your Project.**
2. **Define your Environments.**
3. **Create your Deployment Profiles.**
4. **Deploy your Project.**

These implementation steps are all accomplished using Enterprise Designer, which is introduced in **Enterprise Designer** on page 39 and developed further in subsequent chapters.

2.3 System Architecture

SeeBeyond's eGate Integrator employs a versatile architecture that is ideally suited to distributed computing environments. As a result, the various components of an eGate Integrator system can reside on the same hardware platform (assuming adequate system resources), or be distributed across several different hardware platforms in the enterprise network. Figure 3 shows an example system implementation that is highly distributed.

Figure 3 Distributed eGate Integrator System



Note: *In this scenario it is assumed that all instances of eGate Integrator are of the same release.*

2.3.1 Repository

The setup, components, and configuration information for the elements of a Project are stored in the Repository. The Repository also stores all of the product binary files that are required at run time by the Logical Hosts. The components and configurations are downloaded to the Logical Host during the initial bootstrap process and as needed after design-time configuration changes are made.

As shown in Figure 3, a single Repository serves the entire enterprise. This common Repository is used for development, testing, and production purposes. Communication between the Repository and other ICAN components can be configured to use either HTTP or HTTPS. The Enterprise Designer and Enterprise Manager clients can communicate with the Repository through a firewall. The Repository makes Web Services available via a UDDI registry.

2.3.2 Run-Time Environments

An eGate Integrator Environment represents the total system required to implement a Project. It consists of a collection of Logical Hosts, capable of hosting components of the ICAN Suite, along with information about external systems involved in the implementation.

- **Logical Hosts**
  Each Environment contains one or more system definitions. Each definition must include one or more **integration servers** such as the SeeBeyond Integration Server, which are the engines that run eGate Integrator Collaborations and eWays, and one or more **message servers** such as the SeeBeyond JMS IQ Manager, which manage JMS topics (publish-and-subscribe messaging) and queues (point-to-point messaging). Each collection of integration servers and message servers, plus additional software modules, comprise what is known as a *Logical Host*.
- **External Systems**
  An external system is a representation of a real, physical system that exists within the specific Environment, with configuration properties for locating and accessing that system.

In the example system shown in Figure 3, the production environment is split across two hardware platforms, each supporting a single Logical Host. Separate environments for development and testing should duplicate the structure of the production environment. The test environment should be supported by hardware similar to that supporting the production environment, to allow performance and load testing to give representative throughput results. The hardware supporting the development environment, however, does not usually have the same performance requirements as that supporting the test and production environments.

An ICAN Project is created within the development environment, then migrated to the test environment, and finally to the production environment. This migration path is a necessary and highly critical practice in implementing a working system.

Note again that there is no requirement for the components shown in Figure 3 to run on separate systems; all could run on a single system, provided that resources (CPU, memory and disk) are sufficient to support concurrent usage.

2.4 User Interfaces

eGate Integrator provides two basic graphical user interfaces (GUIs), each of which addresses a different set of users. Enterprise Manager is an interface used by the entire ICAN Suite, the primary users of which are system administrators. Enterprise Designer is used by personnel who are involved in defining a software system for integrating the various enterprise applications using eGate Integrator and other ICAN Suite products.

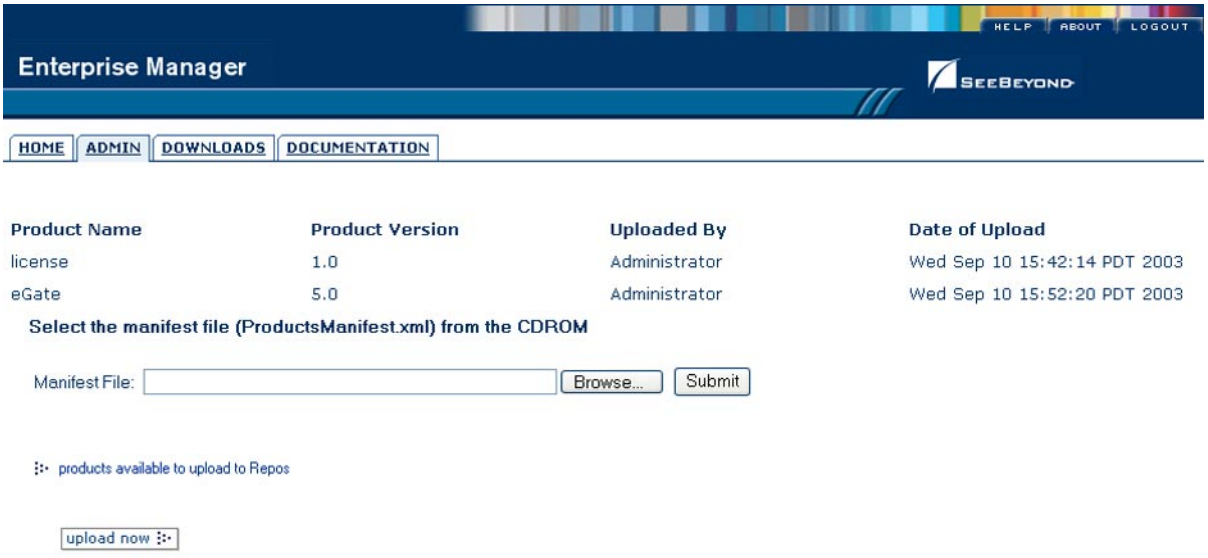
2.4.1 Enterprise Manager

Enterprise Manager is a Web-based application that works within Microsoft Internet Explorer. It is used throughout the SeeBeyond ICAN Suite for:

- Installing and updating ICAN Suite products
- Accessing ICAN Suite product documentation
- Managing and monitoring runtime components

The Enterprise Manager is described in **Enterprise Manager** *on page 42*. Figure 4 shows the Enterprise Manager **Admin** page, used in product installation.

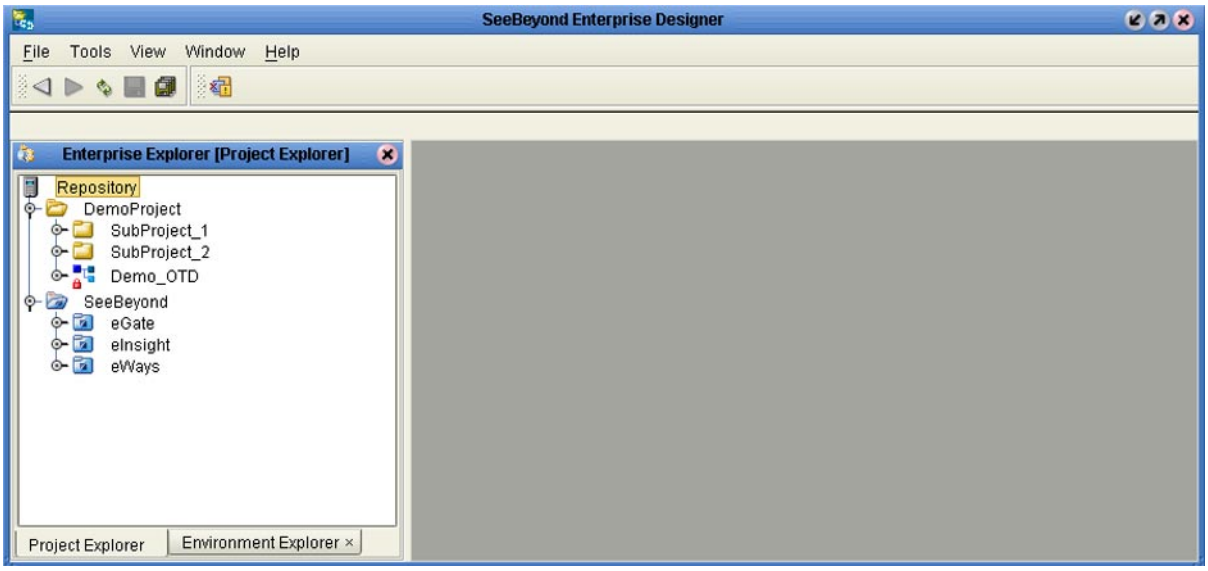
Figure 4 SeeBeyond Enterprise Manager



2.4.2 Enterprise Designer

The SeeBeyond Enterprise Designer is used to create and configure the logical components and physical resources of an ICAN Project. Through this GUI (see Figure 5), you can develop Projects to process and route data through an eGate Integrator system. Enterprise Designer is also used by other components of the ICAN Suite, including eInsight Business Process Manager and eXchange.

Figure 5 Enterprise Designer

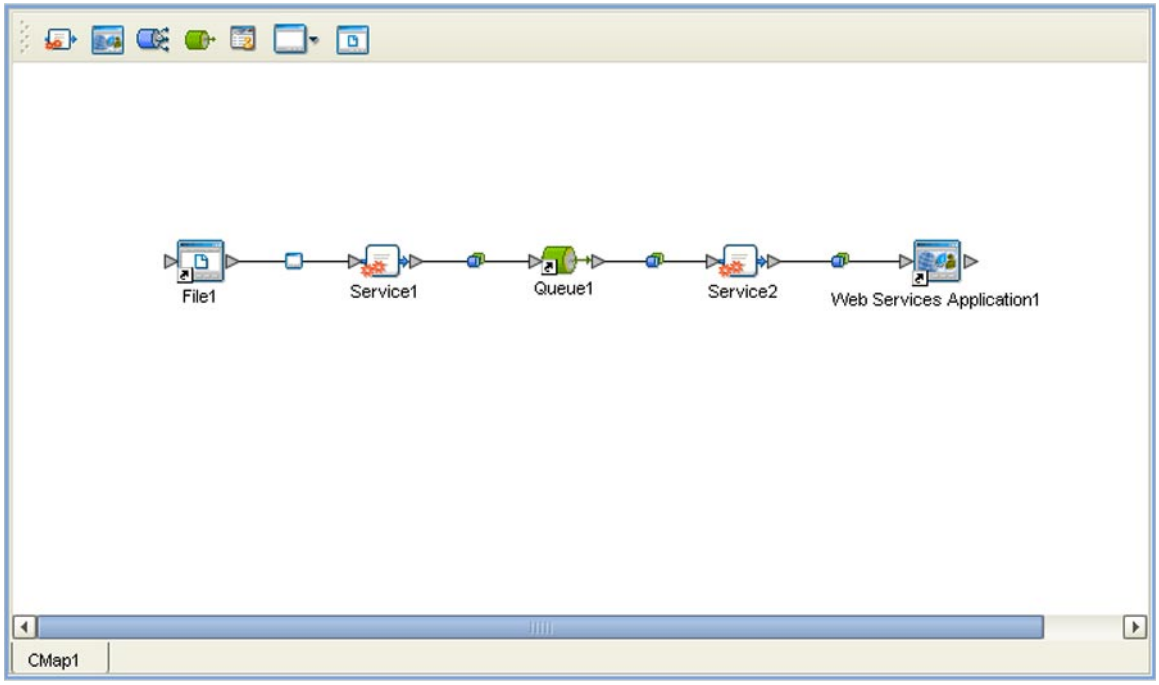


The major features of the Enterprise Designer are the Enterprise Explorer on the left, and an editor panel on the right—which is initially blank. The Enterprise Explorer follows the familiar Windows Explorer format, displaying a tree structure. The editor panel displays a variety of editors, depending upon what component is selected in the Enterprise Explorer. These editors include, for example:

- Connectivity Map Editor
- OTD Editor
- Collaboration Editor (Java)
- Collaboration Editor (XSLT)
- Environment Editor
- Deployment Editor

The Connectivity Map Editor (see Figure 6) provides a graphic example of one of these editors, in which logical components of a Project can be created and connected. eGate Integrator uses Connectivity Maps to intuitively configure the end-to-end flow of messages within an integration. The integration developer can to drag and drop the various Collaborations, Intelligent Queues and external-system eWay adapters onto the Connectivity Map canvas and link them together to specify message flow. The features and usage of the Connectivity Map Editor are described in **ICAN Projects** on page 91.

Figure 6 Connectivity Map Editor



The Enterprise Designer also includes the design-time functionality for other ICAN products, such as eInsight and eXchange. For more information on using other ICAN products in the Enterprise Designer, see the product documentation for those products.

For more information on the Enterprise Designer, see **Enterprise Designer** *on page 49*.

Chapter 3

Enterprise Manager

This chapter provides an introduction to the ICAN Suite Enterprise Manager.

What's in This Chapter

3.1 Overview

Enterprise Manager is a browser-based interface with which you can install and update eGate Integrator, and monitor and manage deployed eGate Integrator components.

Important: *Enterprise Manager works only with Microsoft Internet Explorer.*

3.1.1 Installing and Updating eGate Integrator

eGate Integrator components are uploaded from the installation media (CD-ROMs) to the Repository server via the Enterprise Manager. These products are then available to be downloaded and installed from the Repository server. For information on installing and updating eGate Integrator components, see the *SeeBeyond ICAN Suite Installation Guide*.

3.1.2 Monitoring and Managing eGate Integrator

The Enterprise Manager allows you to monitor and manage deployed eGate Integrator components in real-time.

- **The ICAN Monitor** on page 47 describes the basic features of the ICAN Monitor interface. ICAN Monitor usage for specific tasks is described in the *eGate Integrator System Administration Guide*.
- **The SRE Monitor** on page 48 describes an optional facility that allows you to monitor and manage e*Gate 4.x schemas in eGate Integrator 5.0, using the Schema

Runtime Environment. SRE Monitor usage for specific tasks is described in the *eGate Integrator System Administration Guide*.

3.2 Starting Enterprise Manager

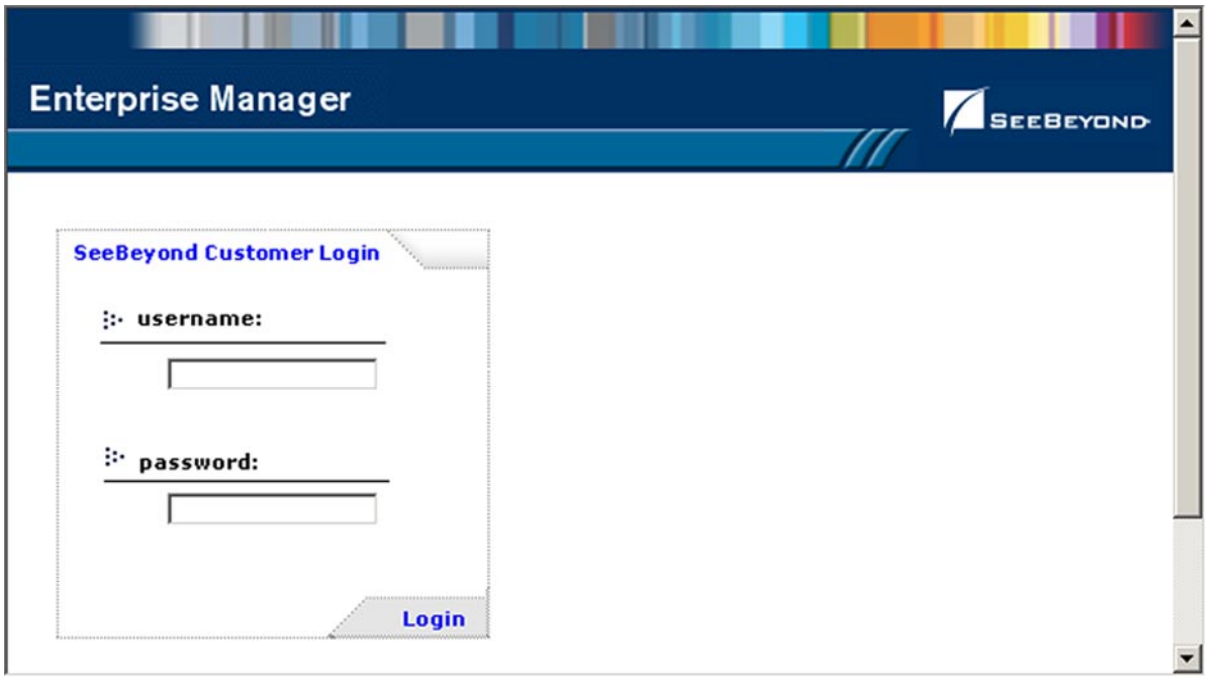
To start the Enterprise Manager

1. Launch Internet Explorer.
2. Enter **http://*hostname*:*portnumber*** in the **Address** box to display the SeeBeyond Customer Login window shown in Figure 7.

Note: *The **hostname** is the fully-qualified, network-addressable host name of the server where you installed the Repository. The **portnumber** is the number of the port you entered during installation of the Repository. See the SeeBeyond ICAN Suite Installation Guide.*

Important: *The TCP/IP host name must be alphanumeric.*

Figure 7 Enterprise Manager Login

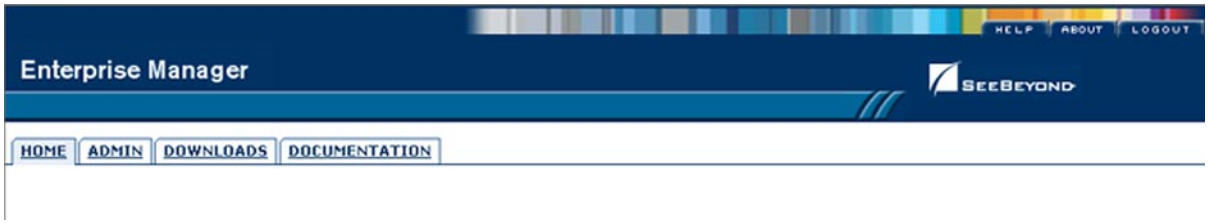


3. Enter your login ID and password in the **Username** and **Password** boxes and click **Login**.

3.3 The Enterprise Manager Interface

Once you have logged in, you see the full Enterprise Manager user interface (see Figure 8).

Figure 8 Enterprise Manager GUI



The Enterprise Manager is organized into four pages, as described in the following table. Each page is accessed by clicking the appropriate tab.

Table 2 Enterprise Manager - Pages

Page	Function
Home	The Home page is used for accessing the ICAN Monitor, which is the main subject of this chapter. See **Home** on page 45.
Admin	The Administration page is used for installing and updating ICAN components. See the *SeeBeyond ICAN Suite Installation Guide* for information.
Downloads	The Downloads page is used in installing and updating ICAN components. See the *SeeBeyond ICAN Suite Installation Guide* for information.
Documentation	The Documentation page is used for accessing ICAN Suite documentation and sample Projects. See **Documentation** on page 46, and the following *Note*.

Note: *You must download the documentation SAR files from the installation disk before you can access any documents using the Documentation page (see the SeeBeyond ICAN Suite Installation Guide).*

There are also three small tabs in the upper-right corner of the Enterprise Manager, which are described in the following table.

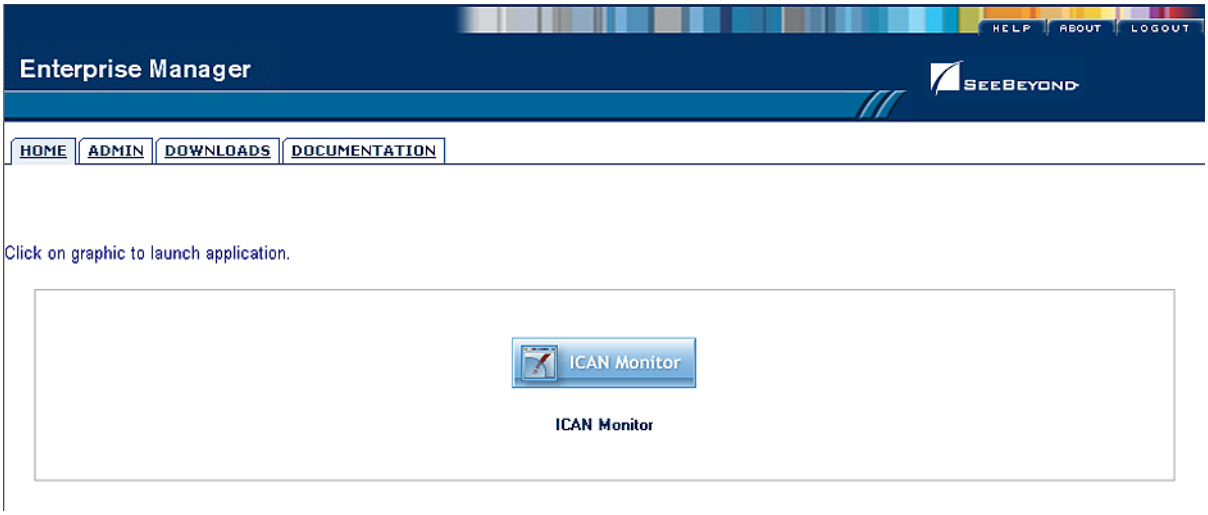
Table 3 Enterprise Manager - Control Tabs

Tab	Function
Help	The Help tab provides access to the online help system.
About	The About tab displays the installed version of the product (this tab is not present on the Documentation page).
Logout	The Logout tab logs you out of the Enterprise Manager and returns you to the Login page.

3.3.1 Home

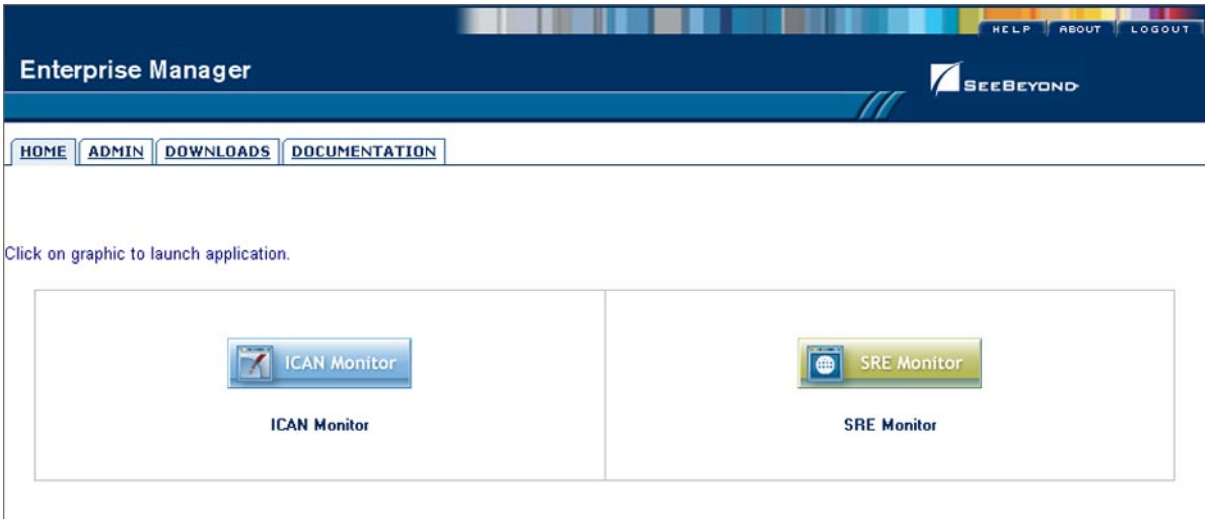
The Enterprise Manager's **Home** page (see Figure 9) contains a link to the ICAN Monitor. Click the **Monitor** icon to launch the ICAN Monitor (see **The ICAN Monitor** on page 47).

Figure 9 ICAN Monitor Launch Icon



If the SRE Monitor (see **The SRE Monitor** on page 48) is installed, its icon is also displayed on this page, as shown in Figure 10.

Figure 10 ICAN Monitor/SRE Monitor Launch Icons

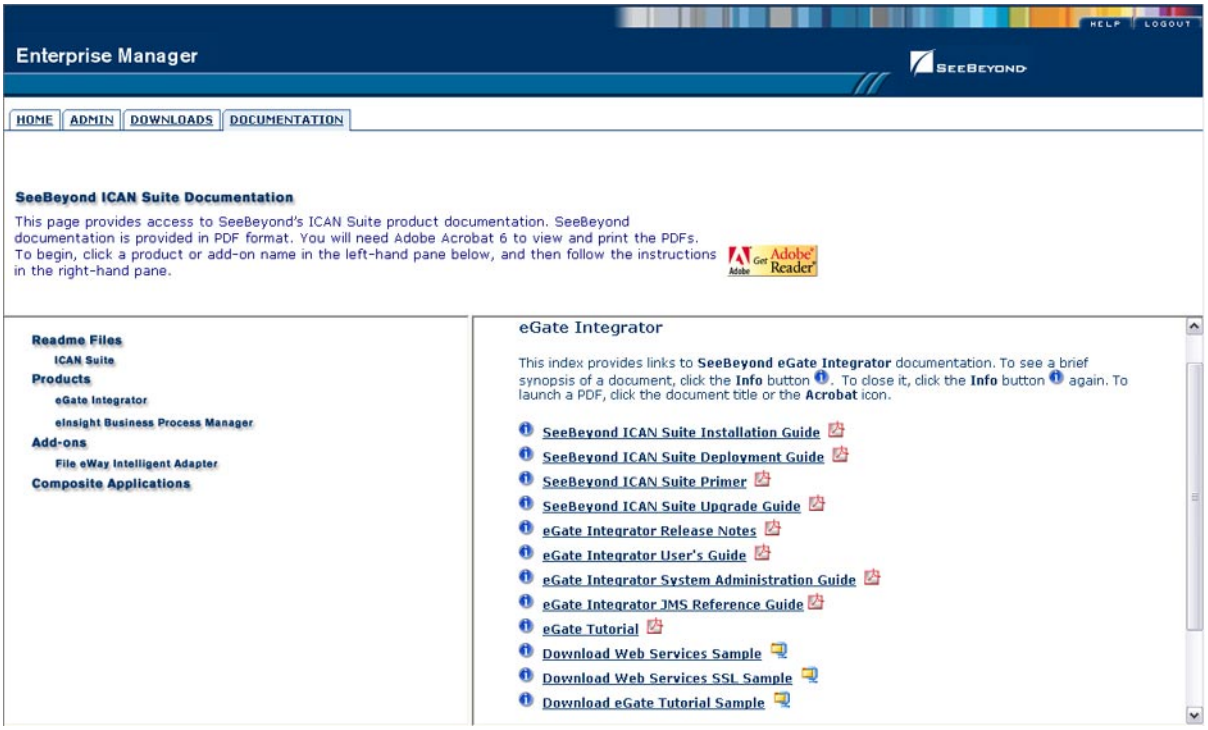


Note: *If connection problems are encountered, close all Internet Explorer windows and retry.*

3.3.2 Documentation

The **Documentation** page (see Figure 11) contains links to the uploaded versions of the SeeBeyond ICAN documentation in Adobe Acrobat (PDF) format, and also any sample Project files (in ZIP format). Shown is the current set for eGate Integrator.

Figure 11 Documentation Page



The provided documentation is organized into the major categories listed in Table 4.

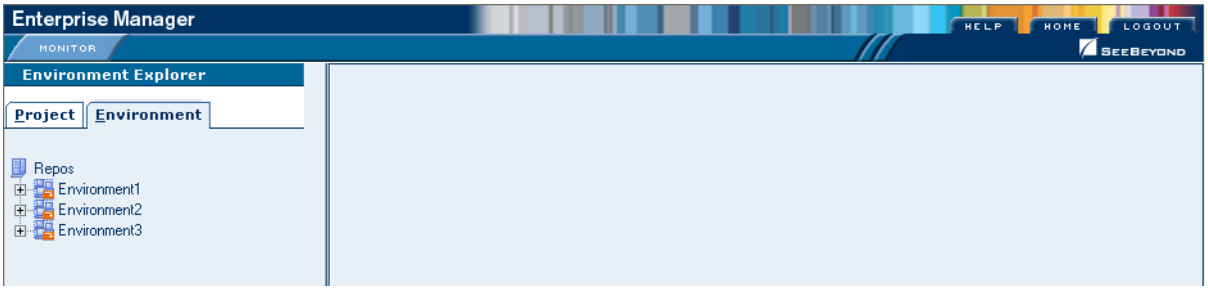
Table 4 Document Categories

Category	Contents
Readme Files	Includes information regarding the latest operating system and hardware requirements, cautions and caveats regarding known issues, and supplementary information arising after the publication of other documentation.
Products	Documentation regarding ICAN core products, such as eGate Integrator and eInsight Business Process Manager. Also includes any installed example Project files.
Add-ons	Documentation regarding optional, ancillary products such as eWays and OTD Libraries.
Composite Applications	Documentation regarding ICAN Composite Applications, such as Corporate Actions or X12 Protocol Manager.

3.4 The ICAN Monitor

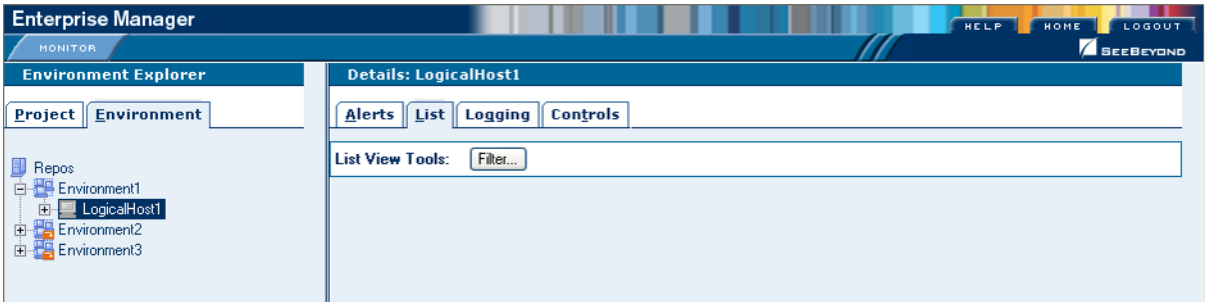
The ICAN Monitor has structure similar to that of the Enterprise Designer, with an Explorer panel on the left and a Details panel on the right. Initially, the Details panel is blank as shown in Figure 12.

Figure 12 ICAN Monitor Interface - Initial



Like the Enterprise Manager itself, the ICAN Monitor's **Details** area is organized into sections represented by tabs (see Table 5). Which tabs are present depends upon the component selected in the Explorer. For example, selecting the Logical Host displays the Monitor page shown in Figure 13.

Figure 13 ICAN Monitor Interface - Environment



At times, the Details panel will have two parts, to display an additional level of information. In this case, different tabs will be displayed in the upper and lower panels.

Table 5 ICAN Monitor Interface - Details Tabs

Tab	Function
Alerts	Displays all alerts for the component selected in the Explorer.
List	Displays a list presenting information about the component selected in the Explorer.
Logging	Displays all log messages for the component selected in the Explorer.
Controls	Displays controls that allow an Administrator to intervene in the run-time process and perform tasks such as starting and stopping components.

See the *eGate Integrator System Administration Guide* for detailed information regarding Monitor usage.

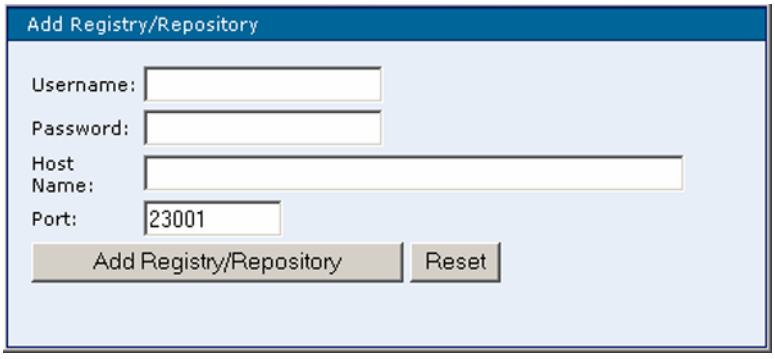
3.5 The SRE Monitor

eGate Integrator 5.0 provides a completely different operating environment from earlier versions of the product (e*Gate). The Schema Runtime Environment (SRE) allows you to use schemas developed for e*Gate 4.x with eGate Integrator 5.0 by providing the necessary environmental components. Instructions for installing and using the SRE are contained in the SeeBeyond documentation for the SRE.

The SRE Monitor enables you to manage e*Gate 4.x schemas running in the Schema Runtime Environment from within eGate Integrator 5.0. The SRE Monitor interface generally resembles the ICAN Monitor, but differs somewhat in detail (see Figure 14). Only the Environment Explorer is displayed, which has two additional icons in the upper left corner; these are described in Table 6.

Figure 14 SRE Monitor

Table 6 SRE Monitor Explorer Icons

Icon	Function
	The **Add Registry/Repository** icon displays the **Add Registry/Repository** dialog box, in which you specify the desired Registry or Repository's name and port, and your user name and password. 
	The **Refresh Registry** icon refreshes the SRE Registry and the Explorer tree following changes to component status.

Instructions for installing the SRE Monitor are contained in the *SeeBeyond ICAN Suite Installation Guide*. See the *eGate Integrator System Administration Guide* for detailed information regarding Monitor usage.

Chapter 4

Enterprise Designer

This chapter presents an overview of the major features of the Enterprise Designer.

What's in This Chapter

4.1 Overview

4.1.1 User Interface

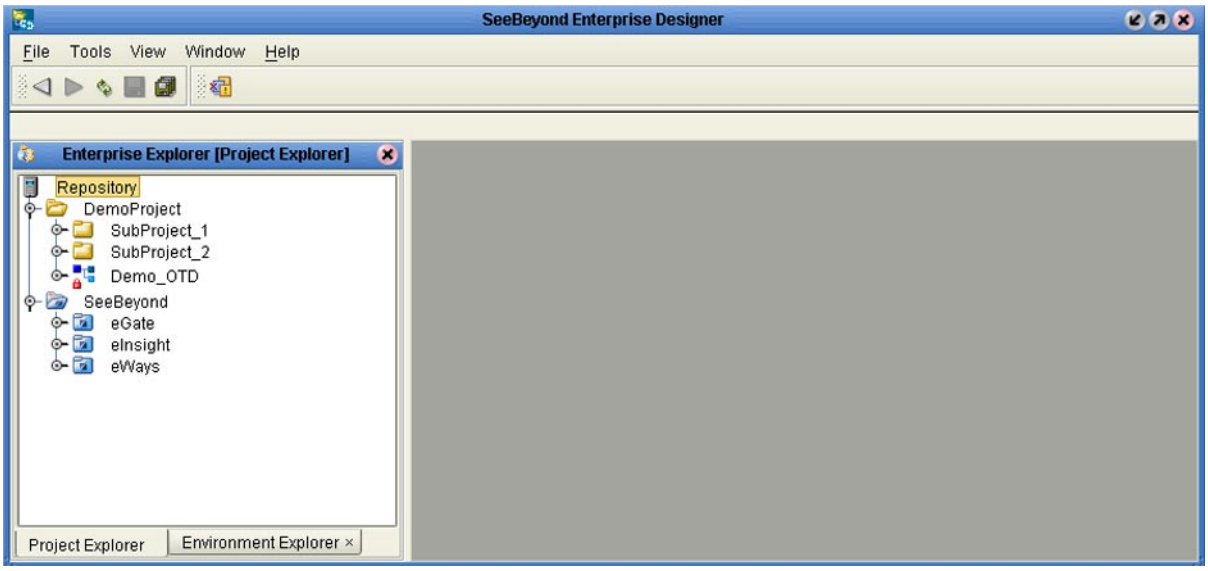
The Enterprise Designer graphical user interface is used to create and configure the logical components and physical resources of an ICAN Project. Enterprise Designer contains the customary graphical interface features, which are described in the following sections of this chapter.

- **Menus** on page 55 describes the options contained in the individual menus.
- **Toolbar** on page 58 describes the functionality of the toolbar icons.
- **Browser Buttons** on page 59 describes the browser buttons that appear throughout the Enterprise Designer, in various wizards and dialog boxes.

The major features of Enterprise Designer are the Enterprise Explorer on the left, and an editor panel on the right—which is initially blank (see Figure 15). The Enterprise Explorer follows the familiar Windows Explorer format, displaying a tree structure. The Enterprise Explorer provides two views of the ICAN system, which are described in the following sections of this chapter:

- **Project Explorer** on page 60
- **Environment Explorer on page 61**

Figure 15 Enterprise Designer User Interface



4.1.2 Editors

The editor panel displays a variety of editors, depending upon what component is selected in the Enterprise Explorer. These editors are described in the following sections of this chapter:

- **Connectivity Map Editor** on page 62
- **OTD Editor** on page 63
- **Collaboration Editor (Java)** on page 64
- **Collaboration Editor (XSLT)** on page 65
- **Environment Editor** on page 66
- **Deployment Editor** on page 67

4.1.3 Analysis and Archiving tools

Enterprise Designer includes several analysis and archiving tools, which are described in the following sections of this chapter:

- **Project/Environment Import** on page 68, which allows you to import a Project that has been created elsewhere.
- **Project/Environment Export** on page 72, which allows you to export a Project to an external file so that it may be used elsewhere.
- **Impact Analyzer** on page 77, which helps you visualize how a change to one part of a Project would affect the rest of the Project.
- **Version Control** on page 79, which allows you to maintain multiple versions of Project components.

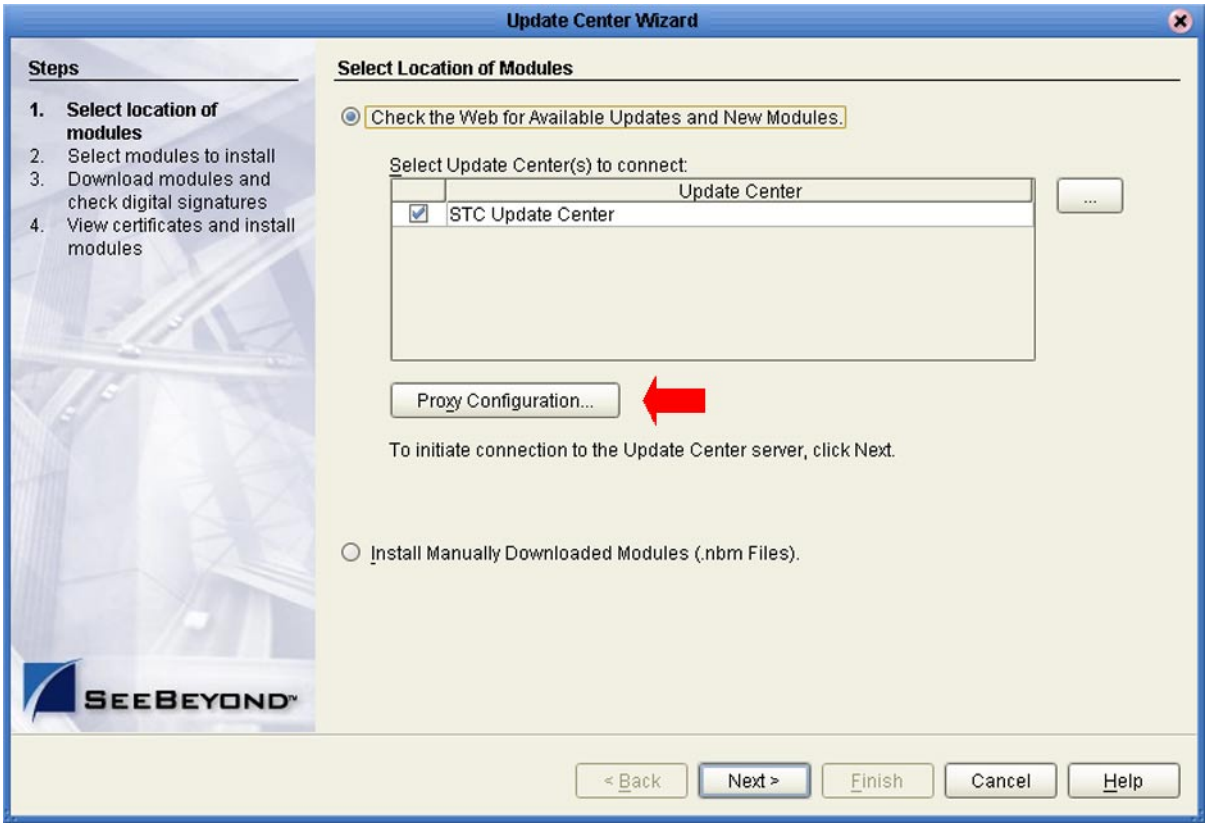
4.2 Using a Proxy Server

The following procedure allows Enterprise Designer to reference a WSDL file using a URL that points outside your firewall. See the *SeeBeyond ICAN Suite Installation Guide* for additional information.

To configure a proxy server

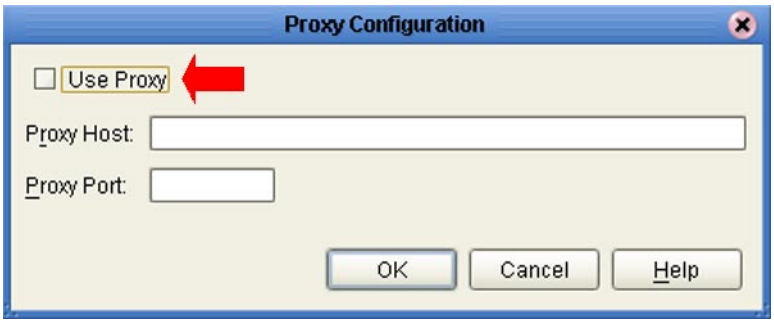
1 From Enterprise Designer's **Tools** menu, select **Update Center** to display the Update Center Wizard (see Figure 16).

Figure 16 Update Center Wizard



2 Click the **Proxy Configuration** button (see Figure 16) to display the Proxy Configuration dialog box shown in Figure 17.

Figure 17 Proxy Configuration Dialog Box



3 Check the **Use Proxy** box, and fill in the fully-qualified name of the proxy host and port for the proxy server.

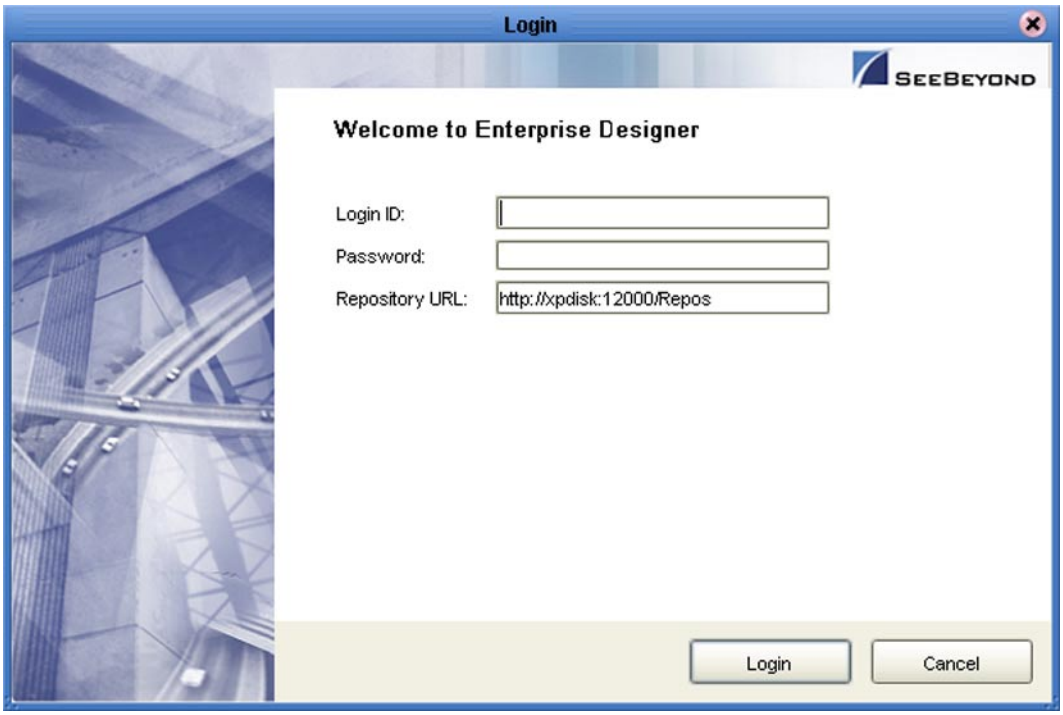
4 Click **OK** to finish.

4.3 Starting Enterprise Designer

To start the Enterprise Designer on a Windows Platform

1 Run the batch file ***ICAN-root*\edesigner\bin\runed.bat** to display the *Login* dialog box shown in Figure 18 (placing a shortcut on your desktop streamlines this procedure).

Figure 18 Login Dialog Box



2 Click in the *Login ID* text box, and enter your login ID.

3 Tab to the *Password* text box, and enter your password.

4 The URL for the Repository should be displayed in the *Repository URL* text box. If it is incorrect, edit the URL before proceeding. See the *SeeBeyond ICAN Suite Installation Guide* for details.

5 Click **Login** to complete the login process and display the Enterprise Designer GUI shown in Figure 15. A progress monitor will appear while the process is running.

To start the Enterprise Designer on a UNIX Platform

1 Run the script ***ICAN-root*/edesigner/bin/runed.sh** to display the *Login* dialog box shown in Figure 18.

2 Click in the *Login ID* text box, and enter your login ID.

3 Tab to the *Password* text box, and enter your password.

4 The URL for the Repository should be displayed in the *Repository URL* text box. If it is incorrect, edit the URL before proceeding. See the *SeeBeyond ICAN Suite Installation Guide* for details.

5 Click **Login** to complete the login process and display the Enterprise Designer GUI shown in Figure 15.

To start the Enterprise Designer on an HP NonStop (HNS) Platform

1 Open a command prompt and change directories to ***ICAN-root*/edesigner/bin**, and type the following command:

```
runed hostname port rep_name
```

where:

- ***hostname*** is the TCP/IP host name of the server where you installed the Repository—not the name of the Repository itself.
- ***port*** is the port number of the Repository.
- ***rep_name*** is the name of the Repository.

This displays the dialog box shown in Figure 18.

2 Click in the *Login ID* text box, and enter your login ID.

3 Tab to the *Password* text box, and enter your password.

4 The URL for the Repository should be displayed in the *Repository URL* text box. If it is incorrect, edit the URL before proceeding. See the *SeeBeyond ICAN Suite Installation Guide* for details.

5 Click **Login** to complete the login process and display the Enterprise Designer GUI shown in Figure 15.

4.4 Interface Features

4.4.1 Menus

The menu bar provides access to a variety of options for managing your Project. The individual menus are described in the following tables.

File Menu

Table 7 File Menu Options

Option	Function
Save	Saves changes to the selected objects (to the local workspace only).
Save All	Saves changes to all objects currently open in the editor (to the local workspace only).
Exit	Closes the Enterprise Designer.

Tools Menu

Table 8 Tools Menu Options

Option	Function
Impact Analyzer	Displays a dialog box in which you can view how one component of a Project impacts other components. See **Impact Analyzer** on page 77.
Options	Displays the Options Setup dialog box, in which you can specify selected options such as heap sizes and language extensions. See **Options Setup** on page 57.
Update Center	Displays a series of dialog boxes in which you can check for program updates. See the *SeeBeyond ICAN Suite Installation Guide*.

View Menu

Table 9 View Menu Options

Option	Function
Environment Explorer	Activates the **Environment Explorer** tab on the Enterprise Explorer. See **Environment Explorer on page 61**.
Project Explorer	Activates the **Project Explorer** tab on the Enterprise Explorer. See **Project Explorer** on page 60.

Window Menu

Table 10 Window Menu Options

Option	Function
Cascade	Displays all open windows so that each window slightly overlaps the others in the Project Editor.
Tile	Displays all open windows in a stacked tile pattern.
Horizontal Layout	Displays all open windows from top to bottom.
Vertical Layout	Displays all open windows from left to right.
Minimize All	Minimizes all open windows so that only the title bar displays at the bottom of the Editor canvas.
Restore All	Returns minimized windows to their original position on the Editor canvas.
Close All	Closes all open windows.

Help Menu

Table 11 Help Menu Options

Option	Function
About Enterprise Designer	Displays an information box giving the version number, copyright information, and ICAN Repository connection information.
Contents	Displays the online help for all installed components of the ICAN Suite that operate within Enterprise Designer.
Help Sets	For future use (currently duplicates the Contents option).

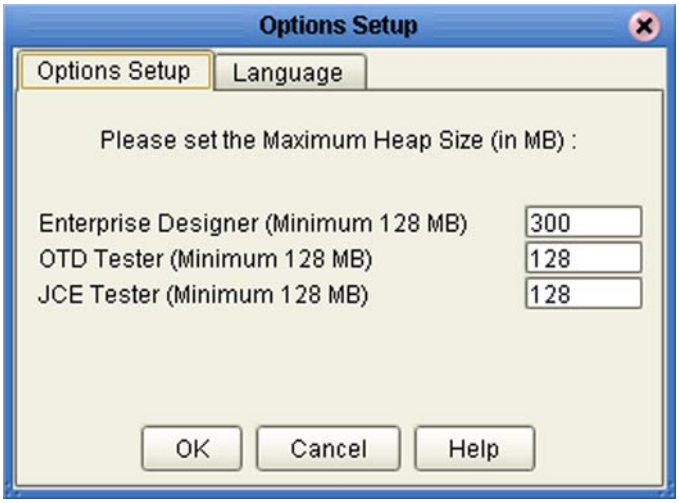
Options Setup

Options Setup

The *Options Setup* tab (see Figure 19) allows you to increase the heap size of Enterprise Designer itself and selected modules to accommodate large file sizes, if necessary.

Note: *If you encounter an out-of-memory error, try increasing the heap size in increments of 50Mb.*

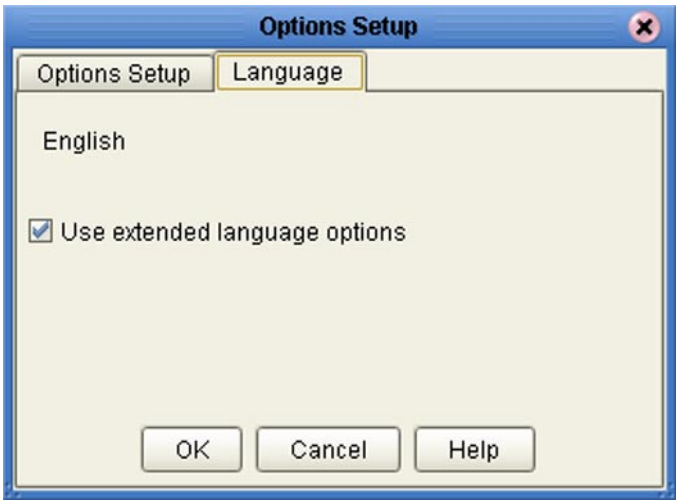
Figure 19 Options Setup - Heap Size Dialog



Language

The *Language* tab allows you to enable extended language options for handling Chinese, Japanese, and Korean characters in the corresponding locale.

Figure 20 Options Setup - Language Dialog



4.4.2 Toolbar

Table 12 Enterprise Designer Toolbar Icons

Icon	Command	Function
	Step Back	Steps back in your usage history for the current session.
	Step Forward	Steps forward in your usage history for the current session.
	Refresh All from Repository	Refreshes the Project Explorer and Environment Explorer to display the current contents of the Repository. (You are prompted to save any changes before the refresh occurs.) Open editors are not refreshed.
	Save	Saves changes made to the selected Project to the local workspace only—the Repository is *not* updated. This icon is inactive if no changes have been made.
	Save All	Saves changes made to all open Projects to the local workspace only—the Repository is *not* updated. This icon is inactive if no changes have been made.
	Impact Analyzer	Displays the *Impact Analyzer* dialog box, which allows you to view how one component of a Project impacts other components.

4.4.3 Browser Buttons

The following buttons are used throughout the Enterprise Designer, in wizards and file selection dialog boxes. They correspond to standard Windows browser buttons.

Table 13 Browser Buttons

Button	Command	Function
	Up One Level	Returns you to the parent folder or directory.
	Home	Returns you to the root folder or directory, or—depending upon the context—the default object.
	Create New Folder	Creates a new folder under the current folder.
	List	Displays folder/file names only.
	Details	Displays details of the folders or files (name, type, date last modified, etc.).

4.5 Enterprise Explorer

The Enterprise Explorer organizes the components of a Project into tabs that display different views of an eGate Integrator system.

- **Project Explorer** on page 60 deals with logical components.
- **Environment Explorer on page 61** deals with physical resources, including the Logical Host and Integration Server.

Note: *The Project and Environment trees are initially loaded only to the Project or Environment level. The contents of a Project or Environment are loaded when you expand the particular node. This causes a slight delay when you expand the node, but eliminates a potentially-significant delay when you open Enterprise Designer, due to the large size of some OTD libraries.*

4.5.1 Project Explorer

The **Project Explorer** tab includes folders and icons that represent the names and contents of Projects. Some example components of a Project are shown in Figure 21.

Figure 21 Enterprise Explorer: Project Explorer View

Details of the features and usage of the Project Explorer are found in **ICAN Projects** on page 91.

4.5.2 Environment Explorer

An Environment consists of Logical Hosts capable of hosting eGate Integrator components and information about external systems which may be involved with an eGate Integrator configuration.

Figure 22 Enterprise Explorer: Environment Explorer View

Details of the features and usage of the Environment Explorer are found in **Environments** on page 344.

4.6 Enterprise Designer Editors

The editor panel—which is initially blank—displays a variety of editors, depending upon what component is selected in the Enterprise Explorer. Additional facilities are also displayed here, such as the Java Debugger (see **Using the Java Debugger** on page 298).

Note: *See the eGate Integrator Tutorial for an end-to-end demonstration of the steps involved in setting up a Project.*

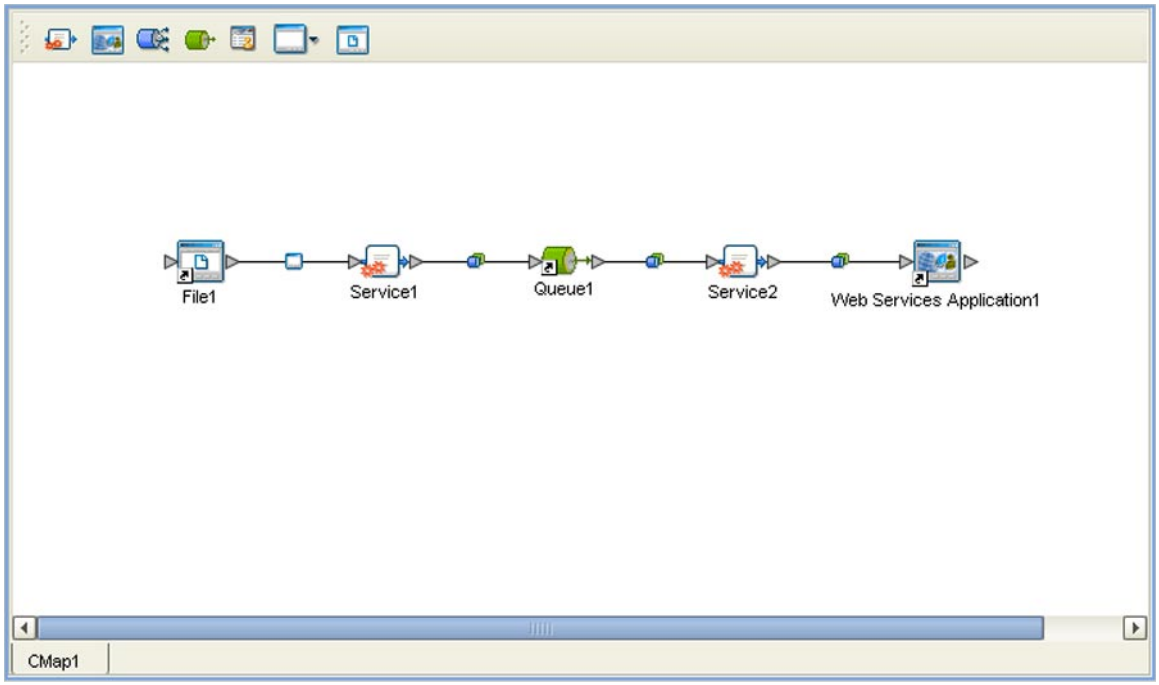
4.6.1 Connectivity Map Editor

A Connectivity Map is a graphical representation of your Project, containing the various logical components comprising the Project and the links between them. The Connectivity Map Editor, shown in Figure 23, allows you to create your Project by simply dragging and dropping icons onto a Project canvas and then connecting them to form data paths. You then can configure the components by means of dialog boxes that are displayed by clicking on the component icons.

Note: *You should create your Collaboration Definitions before using the Connectivity Map to connect components.*

See **Using the Connectivity Map Editor** on page 107 for detailed information.

Figure 23 Connectivity Map Editor

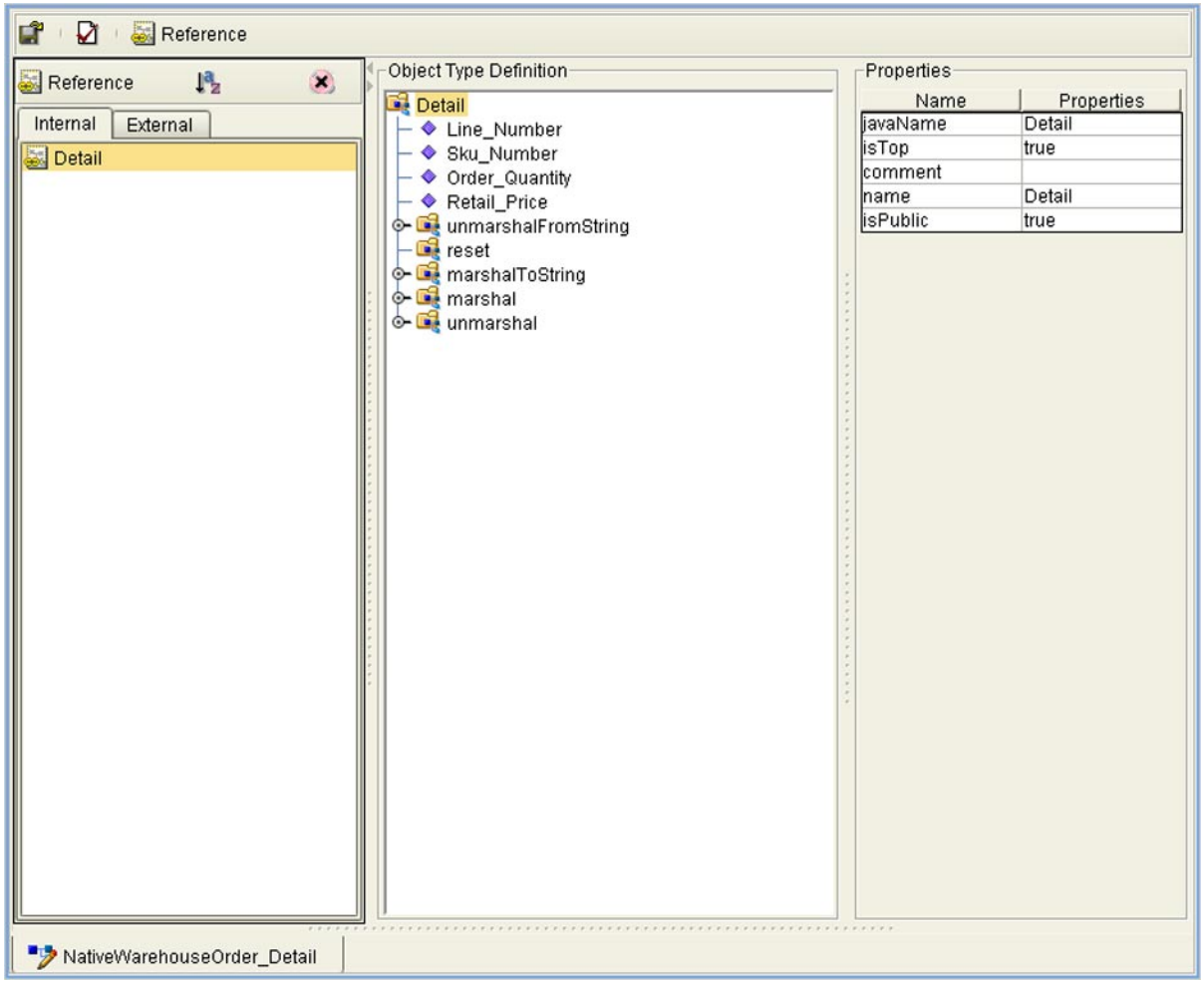


4.6.2 OTD Editor

The OTD Editor window, as shown in Figure 24, displays the source files used to create the Object Type Definitions (OTDs) to use with a Project. You use an OTD wizard tool to create OTD files and add them to the **Project Explorer** tab.

See **OTD Editor** on page 125 for detailed information.

Figure 24 OTD Editor

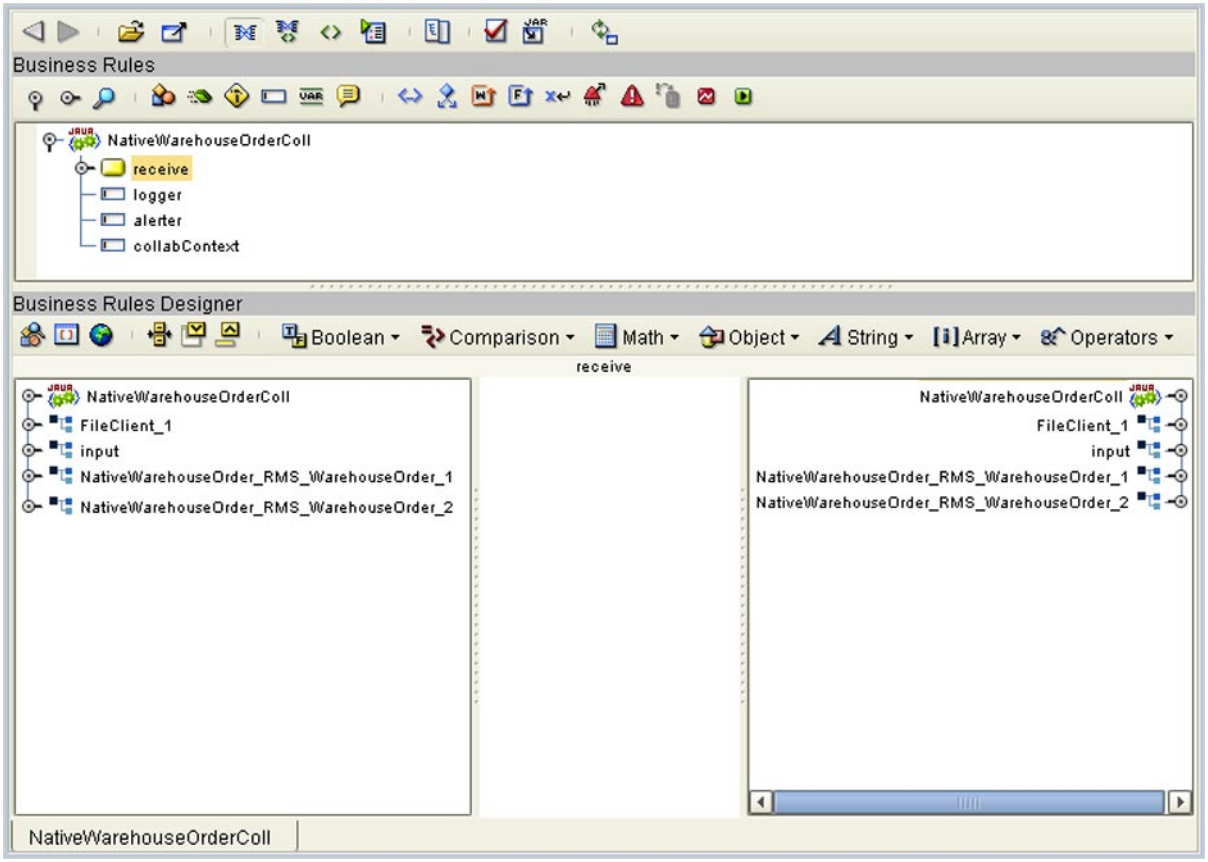


4.6.3 Collaboration Editor (Java)

The Collaboration Editor (Java) window, as shown in Figure 25, displays a Java-based Collaboration Definition that you want to include in a Project. You use a Java wizard tool to create Collaboration Definition files and add them to the **Project Explorer** tab.

See **Using the Collaboration Editor (Java)** on page 193 for detailed information

Figure 25 Collaboration Editor (Java)

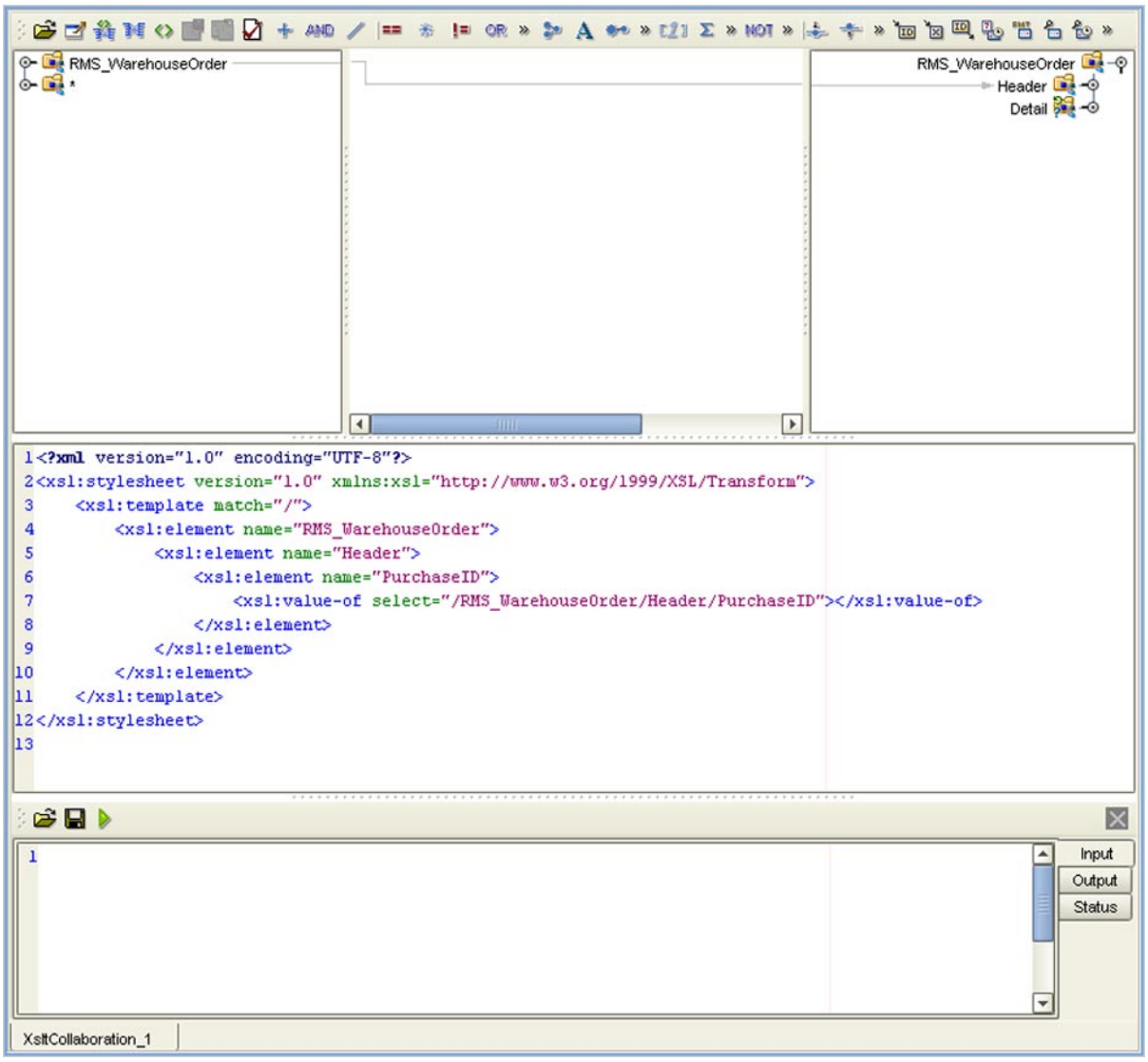


4.6.4 Collaboration Editor (XSLT)

The Collaboration Editor (XSLT) window, as shown in Figure 26, displays the XSLT-based Collaboration Definitions that you need to map together and include in the Project. You use a XSLT wizard tool to create Collaboration Definition files and add them to the **Project Explorer** tab.

See **Using the Collaboration Editor (XSLT)** on page 318 for detailed information

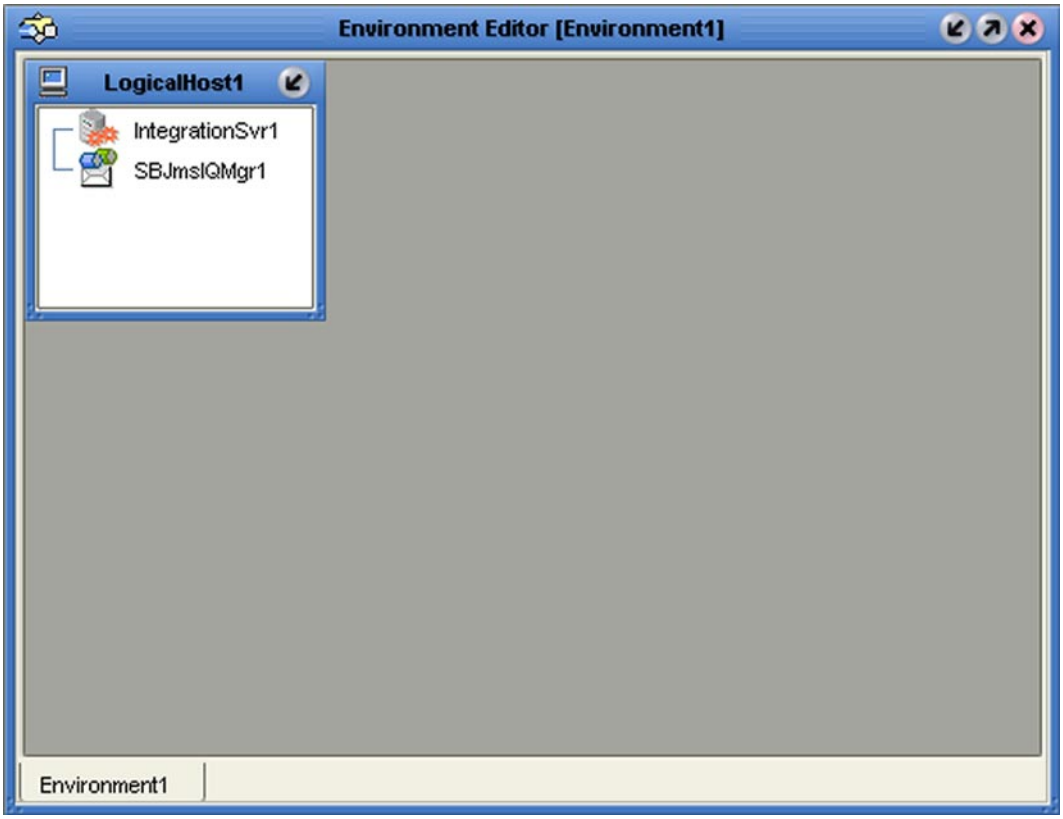
Figure 26 Collaboration Editor (XSLT)



4.6.5 Environment Editor

The Environment Editor provides a canvas in which you can create and customize a run-time Environment. Here you can see the various components (Logical Hosts, servers, and external systems) included in the selected Environment. An Environment containing a populated Logical Host is shown in Figure 27.

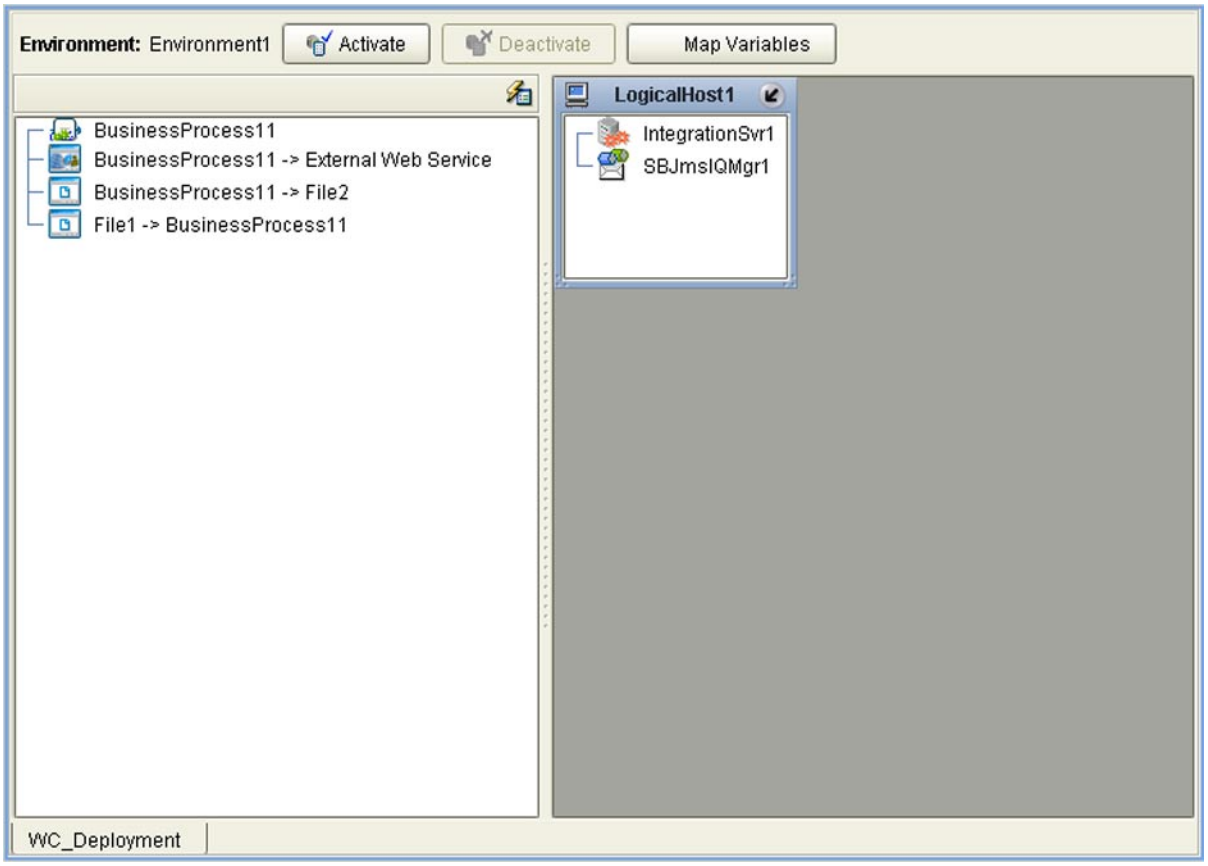
Figure 27 Environment Editor



4.6.6 Deployment Editor

The Deployment Editor, as shown in Figure 28, contains information about how Project components will be deployed in an Environment. See **Deployment Editor** on page 396 for detailed information

Figure 28 Deployment Editor



4.7 Additional Tools and Features

4.7.1 Project/Environment Import

The import function allows you to import an ICAN Project or Environment file using the Enterprise Designer. Both follow essentially the same procedure.

Important: *Products installed in the source Repository must be installed in the Repository into which the Project is imported.*

When importing a Project, note that:

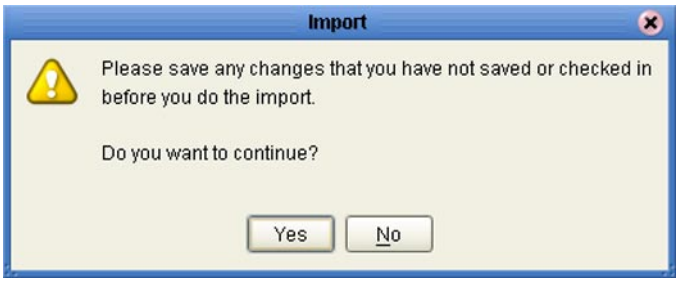
- Existing Projects are not affected by the imported Project.
- During import, if another Project having the same name exists in the target Repository, you will receive an error message and the existing file will not be overwritten.
- If you have not installed all of the necessary products (such as eWays) that a Project requires, you will not be able to import that Project and will get an error message.
- References are validated during import.
- Project deployment objects are not imported (they have references to both Project and Environment elements that are not required at the Project level).

Importing a Project Using Enterprise Designer

To import a Project using Enterprise Designer

1. From the Repository context menu (for Projects) or the Project context menu (for Sub-Projects), select **Import.**
2. The message box shown in Figure 29 appears, prompting you to save your changes.

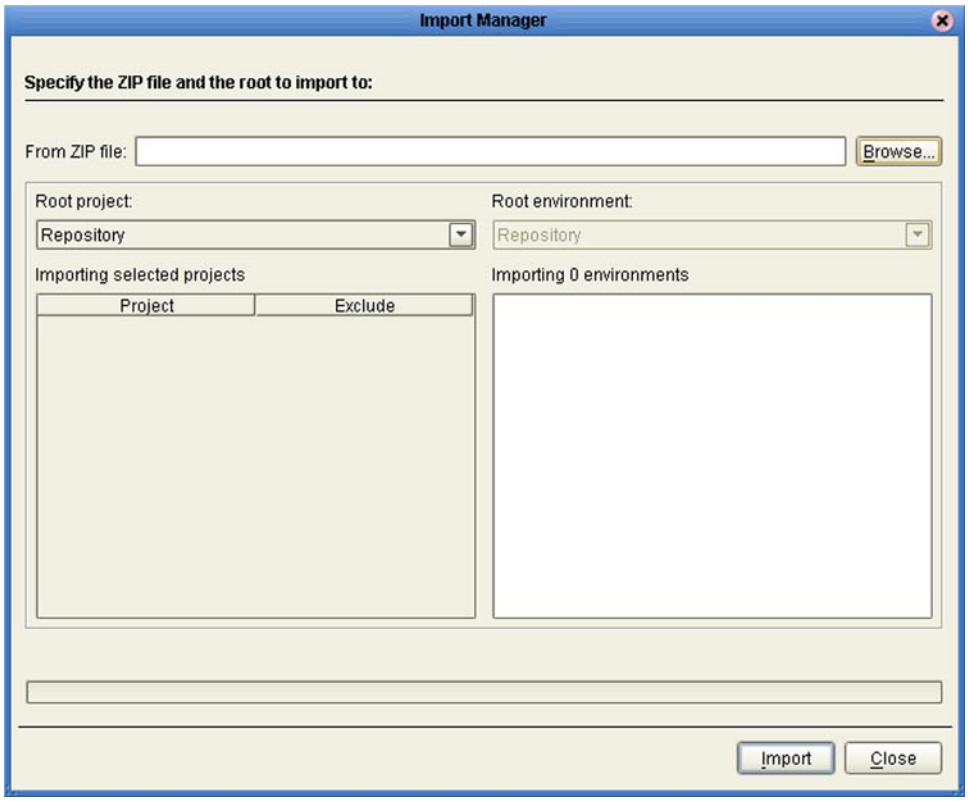
Figure 29 Import Message Box



A. If you want to save your changes, but have not already done so, click **No**. Save your changes, and then re-select **Import**, as in step 1.

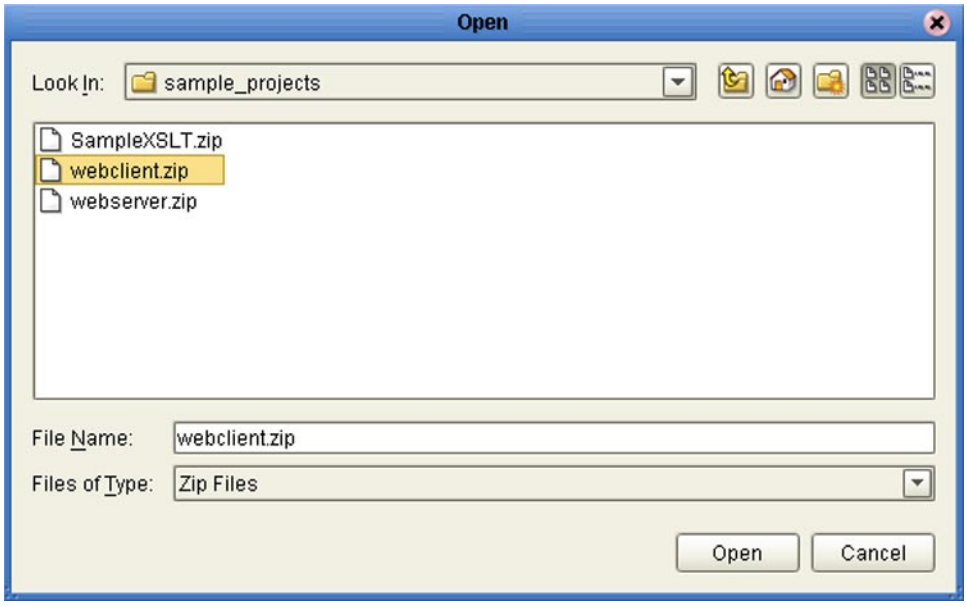
B. If you have saved any desired changes, click **Yes** to display the dialog box shown in Figure 30.

Figure 30 Import Manager Dialog Box (1)



3 Click the **Browse** button to display the *Open File* dialog box, as shown in Figure 31. If you browse to an Environment file, the *Root environment* field will be enabled.

Figure 31 Open File Dialog Box

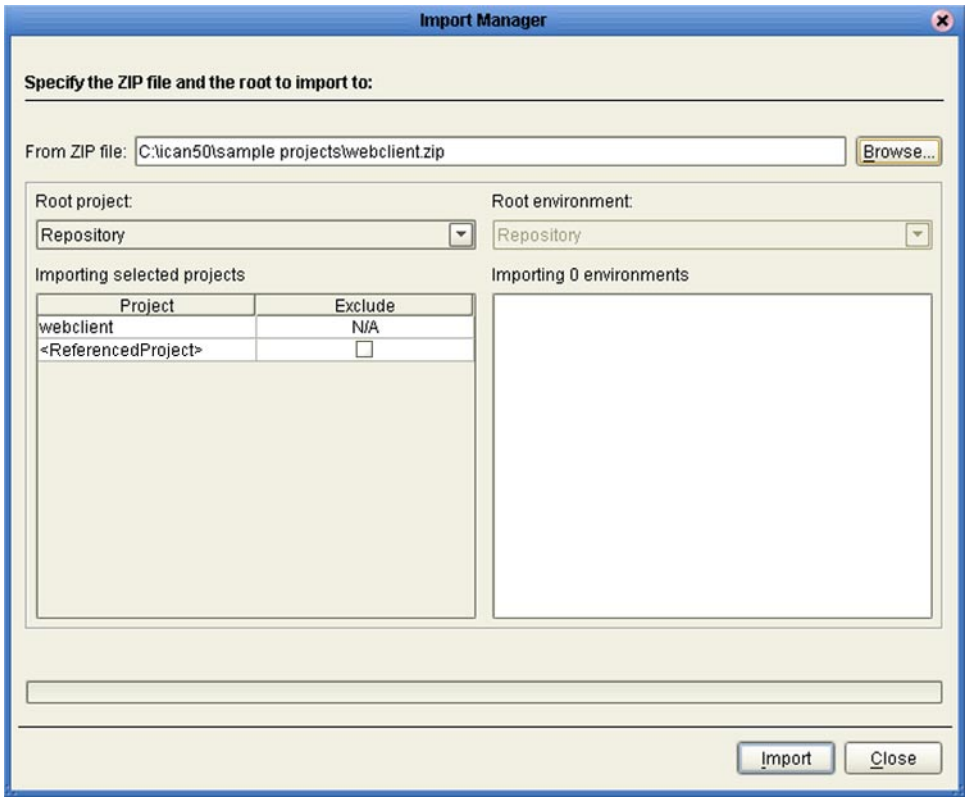


4 Locate and select the Project or Environment file that you want to import.

5 Click **Open** to import the file.

6 The Import Manager dialog box appears as shown in Figure 32.

Figure 32 Import Manager Dialog Box



Note: *If the Project you are importing contains references to another Project, and the other Project already resides in your Repository, you have the option of excluding the referenced Project from the import by checking the box that appears in the Exclude column. The references will be retargeted to the Project existing in the Repository.*

7 Click **Import** to import the file.

8 The Import Status message box shown in Figure 33 appears after the file has been imported successfully.

Figure 33 Import Status Message Box

9 Click **OK** to close the message box.

10 When you are finished importing files, click **Close** to close the Import Manager dialog box. The Project Explorer will now automatically be refreshed from the Repository.

Importing a Project Using the Command Line

You can also import a Project using the following command-line script.

Location of script file:

ICAN-root\repository\util\importProject.bat (or importProject.sh)

Command Syntax:

importProject ***username password importfile rootproject***

where:

- ***importfile*** is the name and path of the archive file containing the Project or Environment you are importing.
- ***rootproject*** is the name of an existing Project, under which the imported Project will become a sub-Project. If the imported Project is not to become a sub-Project, then leave this parameter as an empty string (“”).

Note: *Project names should not include special characters such as a comma (,), single quote (‘), or double quote (“). If any of these characters are used, you must escape them when using the importProject command-line tool.*

To import a Project using the import script

1 Open a command prompt and change directory to ***ICAN-root***\repository\util.

2 To extract a Project contained in the file **c:\project4import.zip** and import it into the Repository, type:

```
importProject username password c:\project4import.zip ""
```

To import a Project as a sub-Project using the import script

1 Open a command prompt and change directory to ***ICAN-root***\repository\util.

2 To extract a Project contained in the file **c:\project4import.zip** and import it into the Repository as a sub-Project of **mainProject**, type:

```
importProject username password c:\project4import.zip mainProject
```

To import an Environment using the import script

1 Open a command prompt and change directory to ***ICAN-root***\repository\util.

2 To extract an Environment contained in the file **c:\environment4import.zip** and import it into the Repository, type:

```
importProject username password c:\environment4import.zip ""
```

3 The Environment, including all Projects deployed to that Environment, will be imported.

4.7.2 Project/Environment Export

The export function allows you to export an ICAN Project and/or Environment to an external file using either the Enterprise Designer or a command-line script.

When exporting a Project, note that:

- The exported Project may have references to elements that are in other Projects. A list of such references is generated during the export process.
- Project deployment objects are not exported, because they have references to both Project and Environment elements that are not required at the Project level.

Exporting a Project Using Enterprise Designer

To export a Project or Environment using Enterprise Designer

1 From the Project context menu, select **Export** to display the Export Manager dialog box. If you do not have any existing Environments in your Repository, you will see the dialog box shown in Figure 34. If you do, you will see the dialog box shown in Figure 35.

Figure 34 Export Manager Dialog Box (1a)

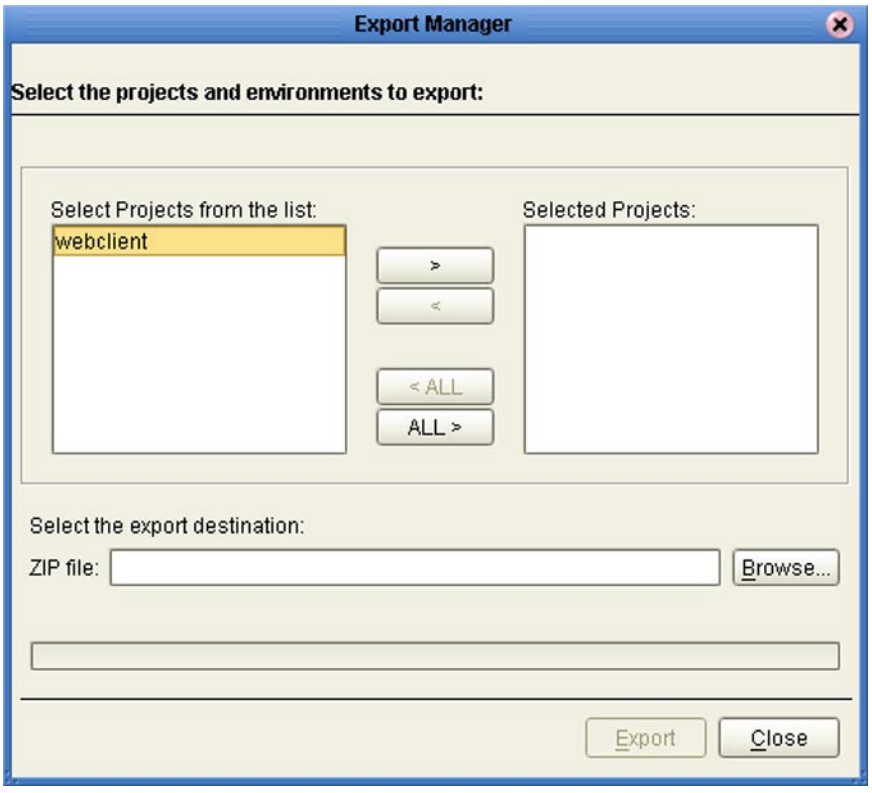
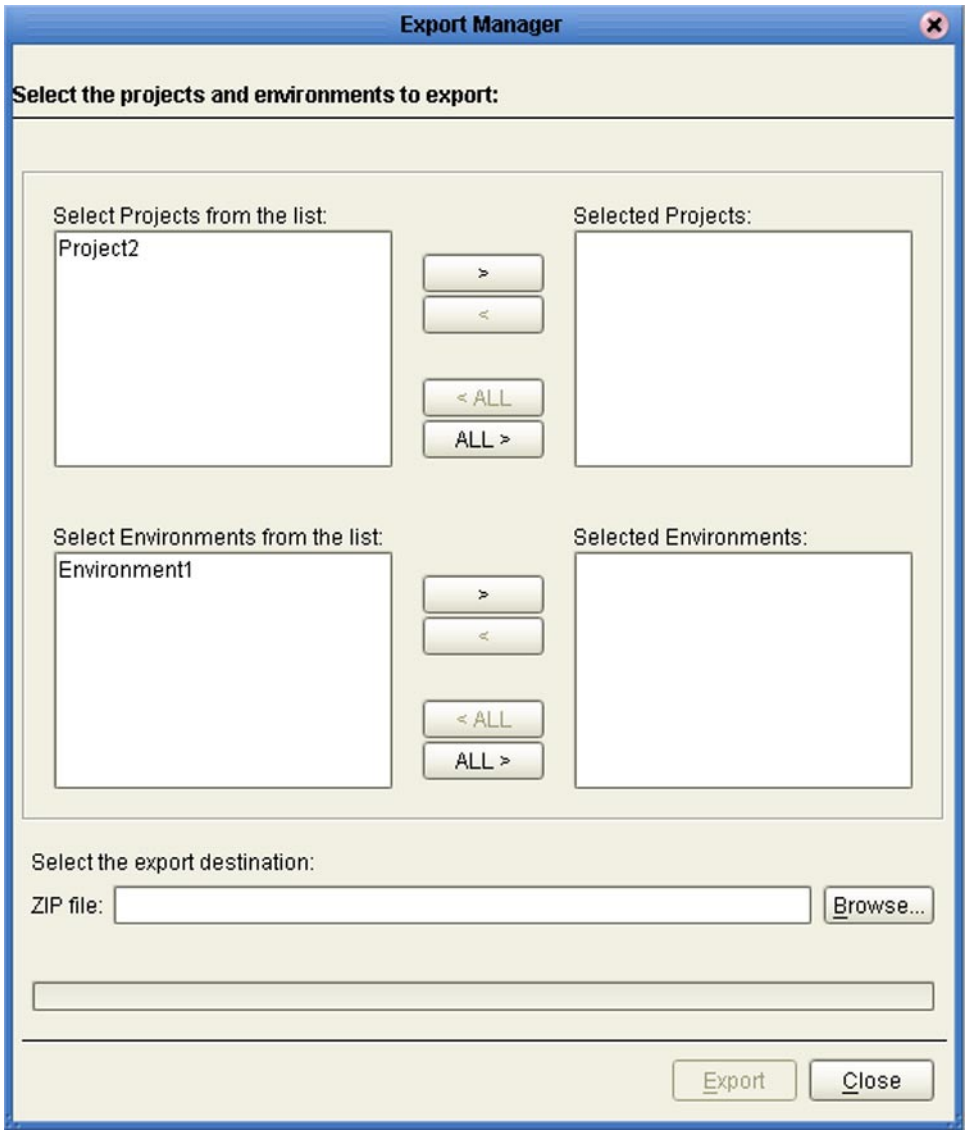
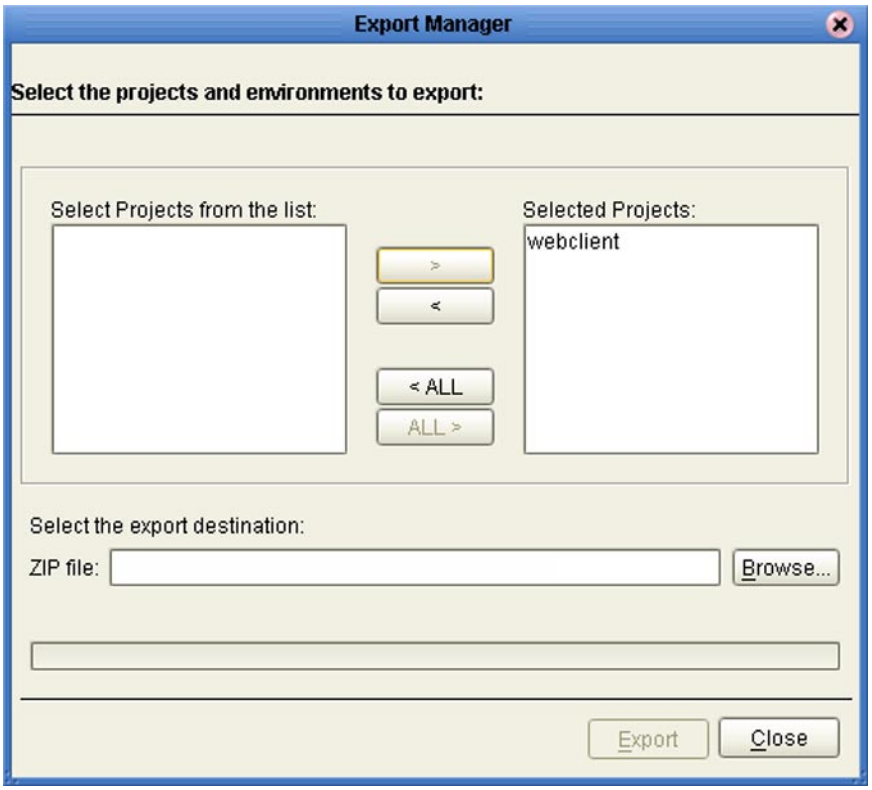


Figure 35 Export Manager Dialog Box (1b)



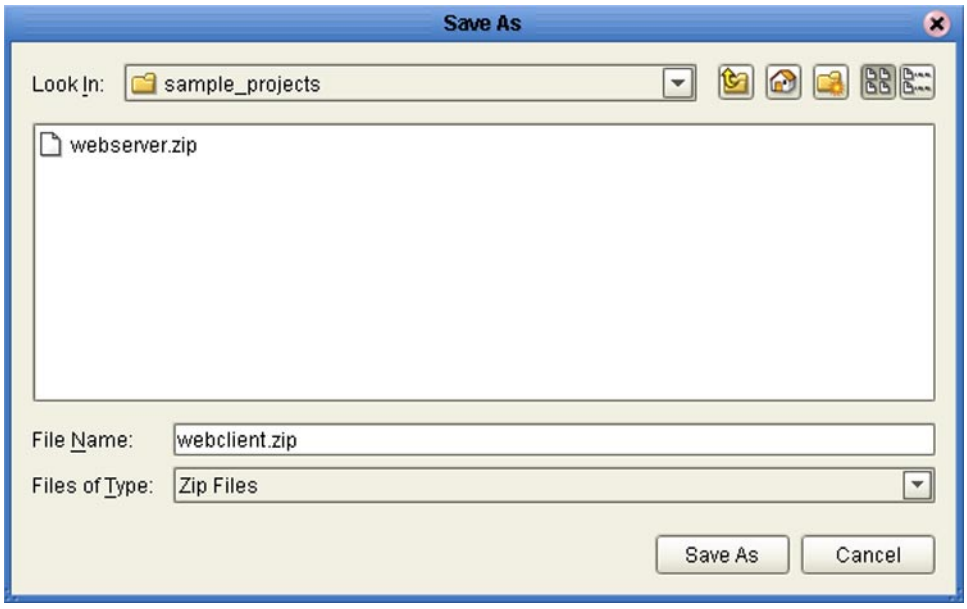
2 Highlight the desired Project(s) or Environment(s) in the displayed list, and transfer them to the *Selected Projects* or *Selected Environments* panel using the arrow buttons (see Figure 36).

Figure 36 Export Manager Dialog Box (2)



3 Click the **Browse** button to display the *Save As* dialog box, as shown in Figure 37.

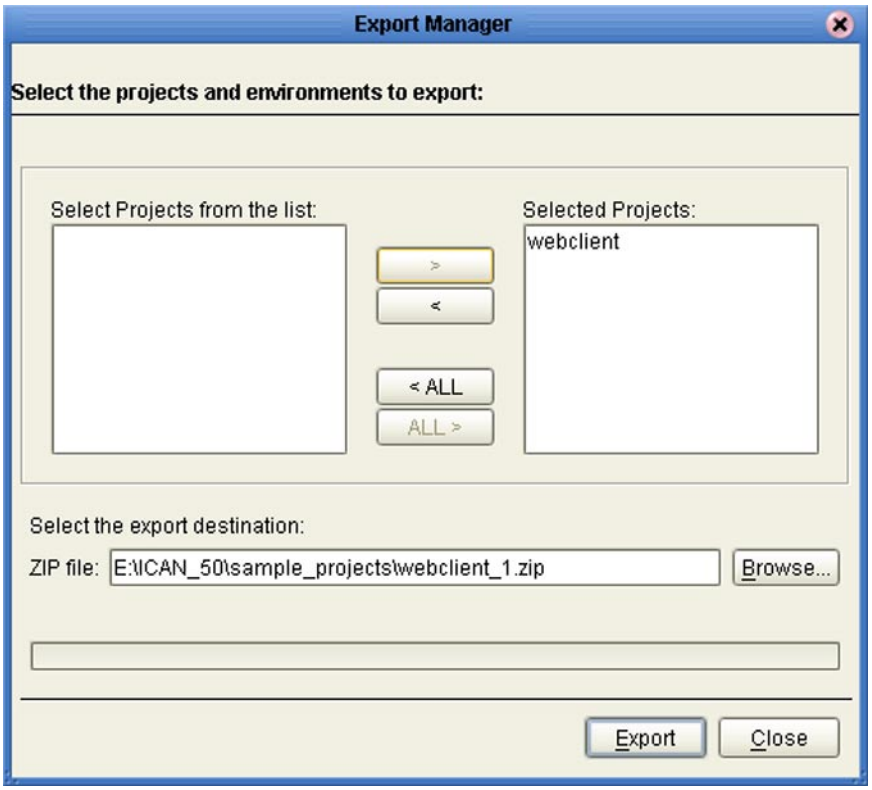
Figure 37 Save As Dialog Box



4 Select the export destination and change the export file name, if desired.

5 Click **Save As** to enter the file name.

Figure 38 Enter File Name Dialog Box (2)



6 Click **Export** to export the Project file (this process may take a few minutes).

7 The Export Status message box shown in Figure 39 appears after the file has been exported successfully.

Figure 39 Export Status Message Box

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

Exporting a Project Using the Command Line

You can also export a Project or Environment using the following command-line script.

Location of script file:

ICAN-root\repository\util\exportProject.bat (or exportProject.sh)

Command Syntax:

```
exportProject username password exportfile projectname
    environmentname
```

where:

- ***exportfile*** is the name and path for the archive file to contain the Project and/or Environment you are exporting.
- ***projectname*** is the name of the Project you are exporting. If you are exporting an Environment only, leave this parameter as an empty string ("").
- ***environmentname*** is the name of the Environment you are exporting. If you are exporting a Project only, leave this parameter as an empty string ("")

Note: *Project names should not include special characters such as a comma (,), single quote ('), or double quote (") . If any of these characters are used, you must escape them when using the exportProject command-line tool.*

To export a Project using the export script

1. Open a command prompt and change directory to ***ICAN-root***\repository\util.
2. To save the Project **myProject** to the file **c:\project4export.zip**., type:

```
exportProject username password c:\project4export.zip myProject ""
```

To export an Environment using the export script

1. Open a command prompt and change directory to ***ICAN-root***\repository\util.
2. To save the Environment **myEnvironment** to the file **c:\environment4export.zip**., type:

```
exportProject username password c:\environment4export.zip ""
    myEnvironment.
```

To export a Project and an Environment using the export script

1. Open a command prompt and change directory to ***ICAN-root***\repository\util.
2. To save the Project **myProject** and Environment **myEnvironment** to the file **c:\projenv4export.zip**., type:

```
exportProject username password c:\projenv4export.zip myProject
    myEnvironment.
```

4.7.3 Impact Analyzer

The Impact Analyzer helps you determine how a change to one component of a Project or Environment will affect other components in that Project or Environment.

To perform an Impact Analysis

1. Select a component in either the Project Explorer or Environment Explorer.
2. Click the **Impact Analyzer** button, or select **Impact Analyzer** from the Tools menu, to display the *Impact Analyzer* dialog box shown in Figure 40.
3. In the *Please show me* drop-down list, select items you would like to view. You have the option of viewing either:
   - Objects that have references to the selected object.
   - Objects that are referenced by the selected object.
4. You can filter the number of listed objects using the *Please show me impacted objects in* drop-down list; the default is the entire **Repository**.
5. You can print the object list by clicking **Print** to display the Windows *Print* dialog box.

Figure 40 Impact Analyzer Dialog Box

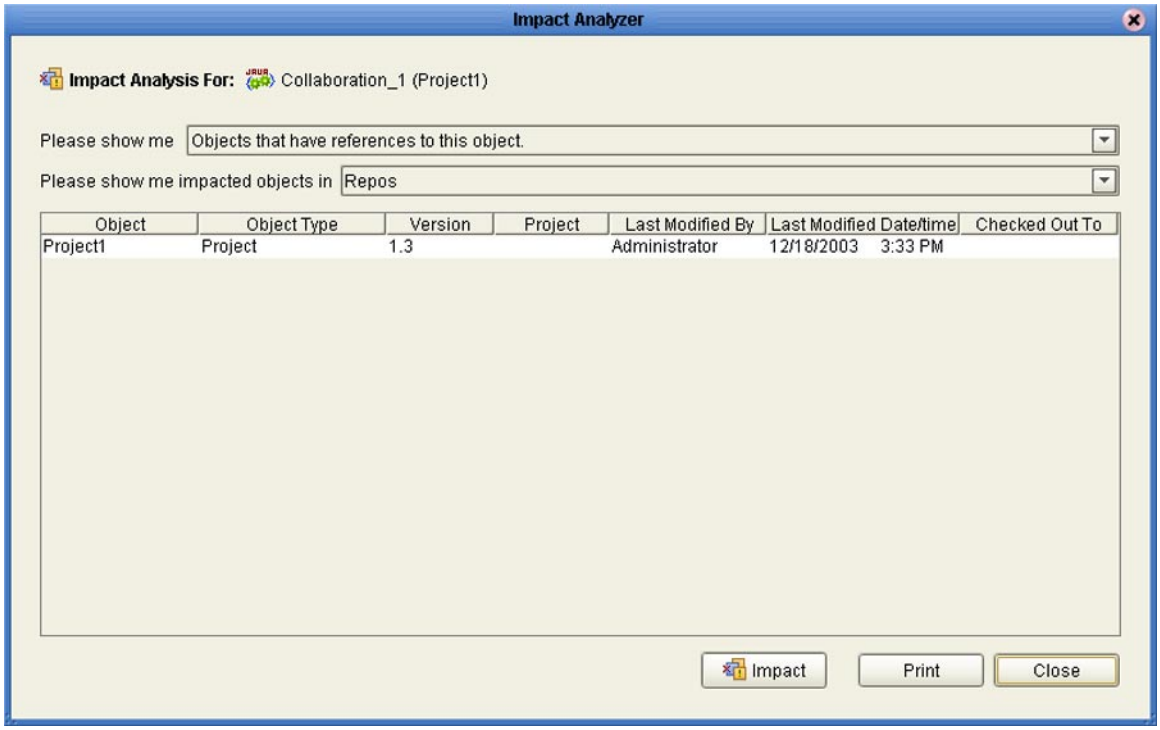
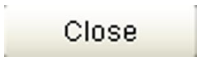


Table 14 Impact Analyzer Command Buttons

Button	Function
	Performs an impact analysis for the object selected from the object list (not fully implemented).
	Displays the Windows **Print** dialog box, which you can use to print the object list.
	Closes the **Impact Analyzer** dialog box.

4.7.4 Version Control

Version control allows you to maintain multiple versions of selected Project or Environment components. The version history of each component is recorded to a log file, and can be viewed by means of a menu option (see **Viewing a Component's Version History** on page 80).

Important: *More than one person concurrently using the same user ID will circumvent this version control system, and one person's work can be overwritten by another. You should ensure that all personnel using Enterprise Designer use unique IDs.*

Checked-In State

When a component is checked in to the version control system, it is locked against modification until being checked out, and a lock is displayed in the component's icon in the Enterprise Explorer. Figure 41 shows the lock, using the OTD icon as an example. See **Checking a Component In** on page 81, **Checking a Component In Without Revisions** on page 82, and **Checking In a Previous Version as the Latest Version** on page 86.

Figure 41 *Checked In* Icon (OTD Example)

Checked-Out State

When the latest version of a component is checked out from the version control system, it is locked against another user checking it out. A writing pad icon (see Figure 42) is displayed next to the component's icon in the Enterprise Explorer, indicating that it is checked out. See **Checking a Component Out** on page 83.

Figure 42 *Checked Out* Icon

Retrieved State

When any version of a component is retrieved from the version history dialog box, it is *not* locked against another user checking it out or retrieving it. A combined writing pad/warning icon (see Figure 43) is displayed next to the component's icon in the Enterprise Explorer, indicating that it is in your workspace—but warning you that it is not locked in any way. See **Retrieving a Component to Your Workspace** on page 84 and **Checking In a Previous Version as the Latest Version** on page 86.

Figure 43 *Retrieved* Icon

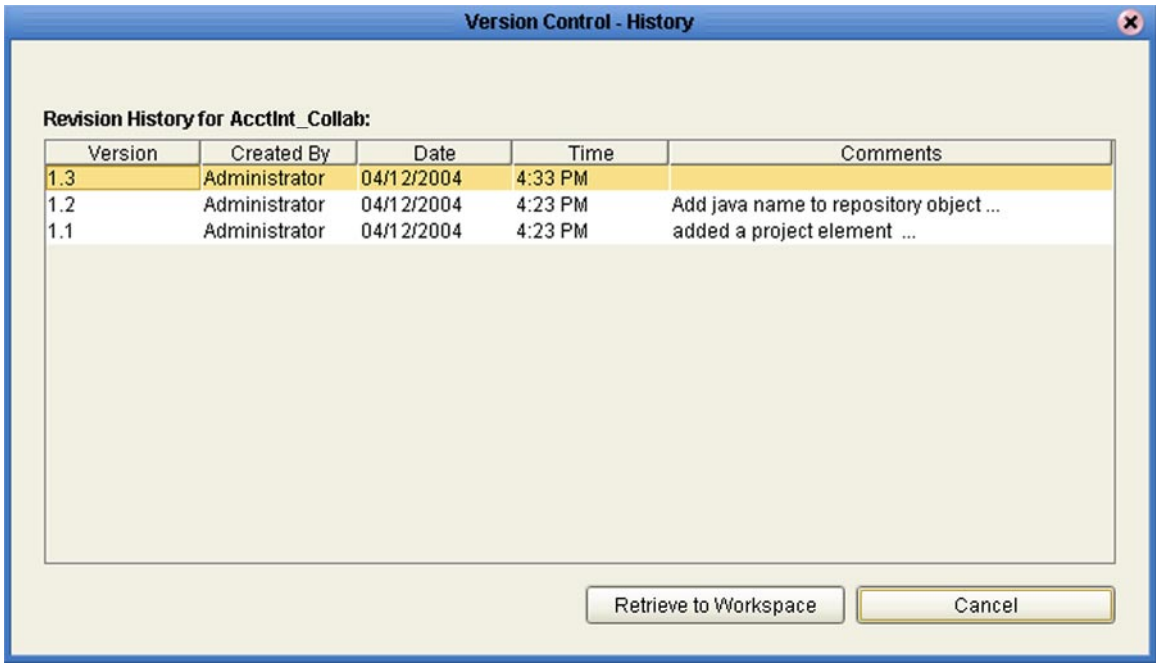
Viewing a Component's Version History

To view the version history for a component

1 In the Enterprise Explorer, select the component and right-click to display its context menu.

2 Select **Version History** to display the *Version Control - History* information box shown in Figure 44.

Figure 44 Version Control - History Dialog Box



3 Double-click in the *Comments* column to display the full text of the comment.

4 Click **Cancel** to close the box.

Note: *If a version is checked out to any user's workspace, or retrieved to your workspace, the appropriate icon also appears in the Version column.*

Note: *The version history for a component that has been **cut** and pasted is preserved, since there can be only one instance of it. The version history for a component that has been **copied** and pasted is **not** preserved, since there can be many instances of it; the version number for each pasted instance is reset.*

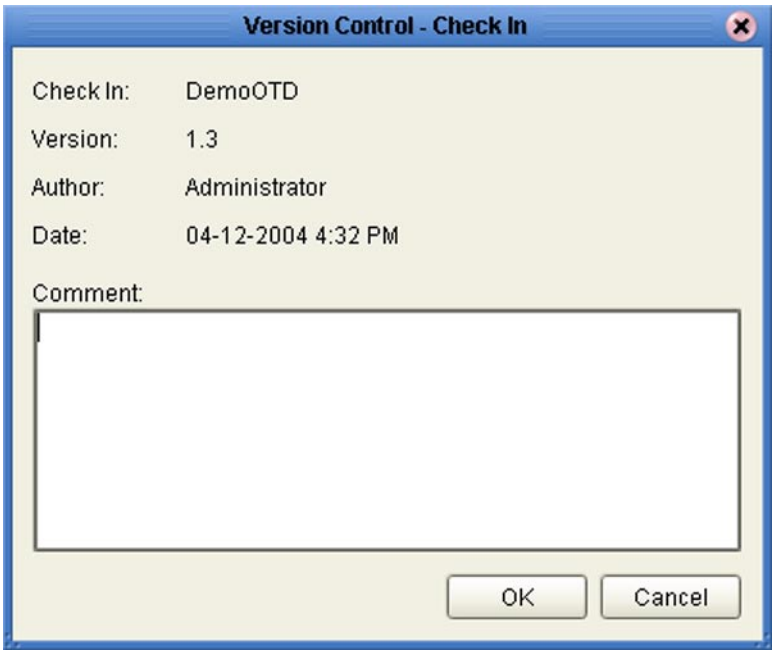
Checking a Component In

Once you have created and configured a component for the first time, or created a revised version of an existing component, you must check that component in to save it to the common area of the Repository and release your lock on the object.

To check in a new version of a Project or Environment component

1. In the Enterprise Explorer, select the component and right-click to display its context menu.
2. Select **Check In** to display the *Version Control - Check In* dialog box shown in Figure 45.

Figure 45 Version Control - Check In Dialog Box



3. Type in a description of the changes in the new version.
4. Click **OK** to check the new version in.

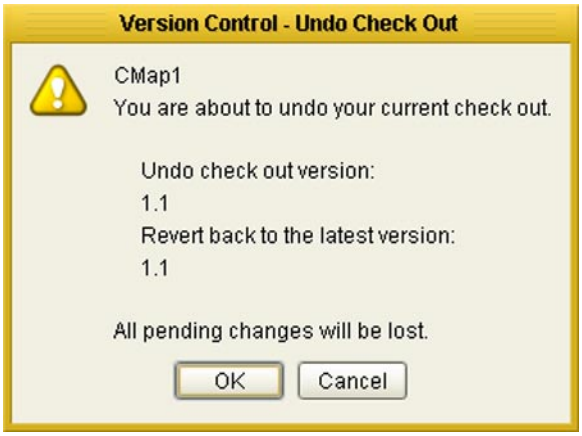
Checking a Component In Without Revisions

When you have checked the latest version of a component out and want to check it back in without any revisions, you can simply cancel the check-out by using the following procedure. The version number is not incremented.

To check in a Project or Environment component without revisions

1. In the Enterprise Explorer, select the component and right-click to display its context menu.
2. Select **Undo Check Out** to display the *Version Control - Undo Check Out* dialog box shown in Figure 46. (This option is not available for all components.)

Figure 46 Version Control - Undo Check Out Dialog Box



3. Click **OK** to check the currently checked-out version back in.

Note: *This procedure is also valid for retrieved versions.*

Checking a Component Out

You can check out the latest version of a component for editing by using the following procedure.

Note: *Only one user can have a file checked out for editing at a time. If another user attempts to check out the same file, they will receive a message indicating that the file is currently checked out.*

To check out the latest version of a Project or Environment component

1. In the Enterprise Explorer, select the component and right-click to display its context menu.
2. Select **Check Out** to display the *Version Control - Check Out* dialog box shown in Figure 47.

Figure 47 Version Control - Check Out Dialog Box

3. Click **OK** to check the component out.
4. Click the **Save** or **Save All** icon to place the version in your Repository workspace.

Note: *Checking out a Java Collaboration disables the cut-and-paste feature for other users.*

Retrieving a Component to Your Workspace

You can retrieve either the current or a previous version of any of the following components by retrieving it from the Version History information box.

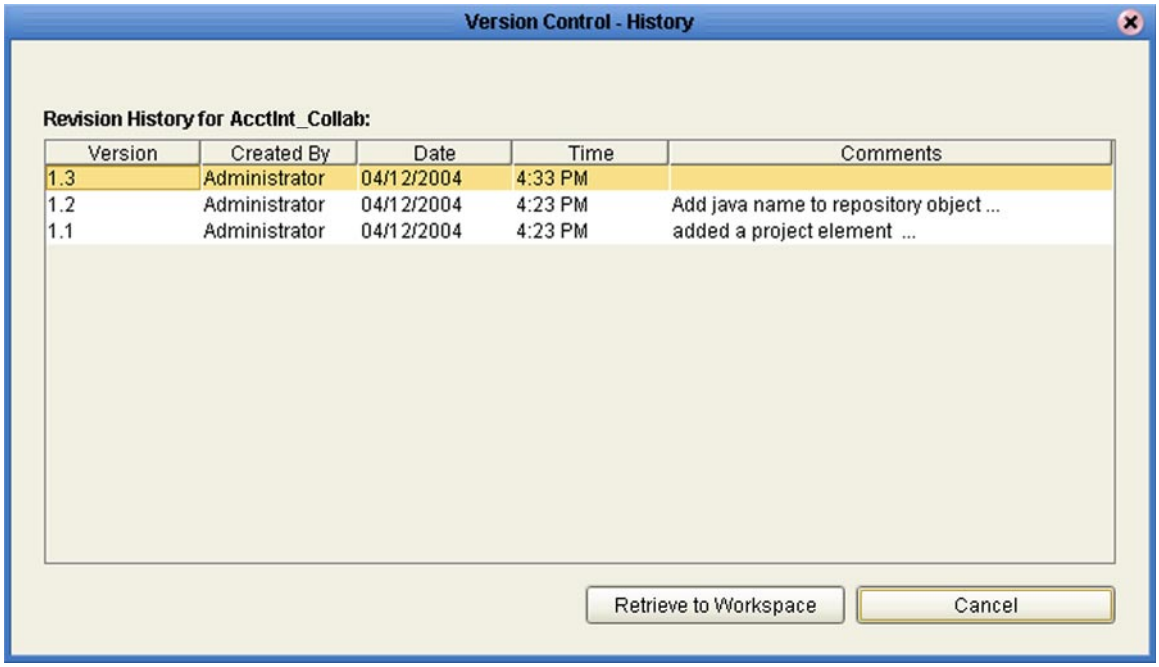
- Collaboration Definition (Java)
- Object Type Definition (DTD)
- Object Type Definition (XSD)
- Object Type Definition (User-Defined)
- eInsight Business Process

Retrieving does *not* lock the file from being checked out or retrieved by other users for editing. To check a retrieved version back in as the latest version, you must use the **Make Latest** option described in **Checking In a Previous Version as the Latest Version** on page 86.

To retrieve an older version of a Project or Environment component

1. In the Enterprise Explorer, select the component and right-click to display its context menu.
2. Select **Version History** to display the *Version Control - History* dialog box shown in Figure 44.

Figure 48 Version Control - History Dialog Box



3. Select the version you want to retrieve and click **Retrieve to Workspace**.
   A. If you are attempting to retrieve the *latest* version of the component, you will be presented with the dialog box shown in Figure 49.

Figure 49 Access File Dialog Box

- **Check Out for Edit** copies the file to your workspace and locks it—the file becomes read-only to other users. This is the same mechanism as described in **Checking a Component Out** on page 83. To check the latest version out from the dialog box, you must select this option and click **OK**. You will then be presented with the dialog box shown previously (Figure 47).
- **Retrieve to Workspace** copies the file to your workspace, but does not prevent it from being checked out or retrieved by other users. This is the default setting for the dialog box; simply click **OK**.

B If you are attempting to retrieve a *previous* version of the component, you will be presented with the dialog box shown in Figure 50. Clicking **OK** will overwrite any other version you have retrieved to your workspace, or replace the currently checked-in version in your workspace only—other users will be unaffected.

Figure 50 Confirm Version Replace Dialog Box

Note: *If you have the latest version of the component checked out to your workspace, the* ***Checked Out*** *icon will appear in the Version column of the Version History dialog and the* ***Retrieve to Workspace*** *button will be disabled. You must check the latest version back in to version control before you can retrieve any version.*

Checking In a Previous Version as the Latest Version

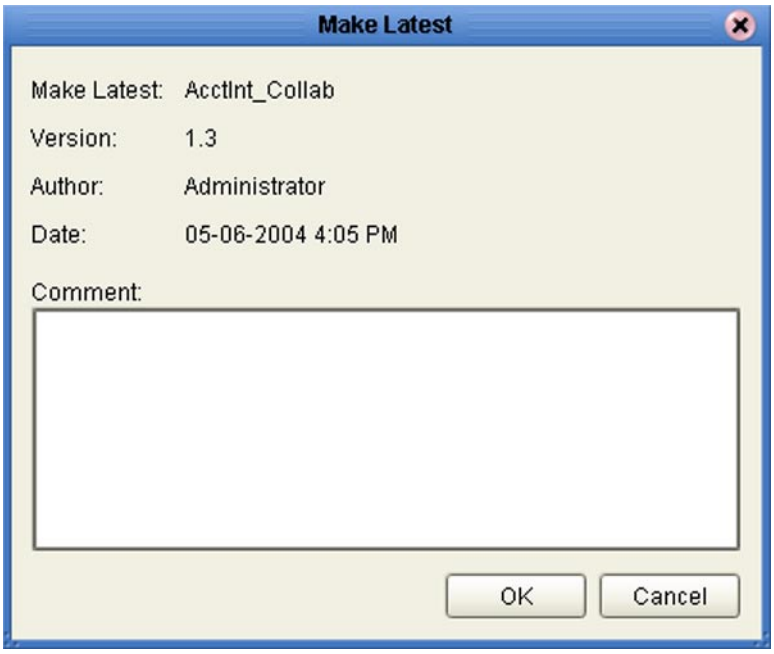
If you have retrieved a previous version of any of the following components to your workspace, you can check it in to the version control system as the latest version by selecting the *Make Latest* option.

- Collaboration Definition (Java)
- Object Type Definition (DTD)
- Object Type Definition (XSD)
- Object Type Definition (User-Defined)
- eInsight Business Process

To make a previous version of a component become the latest version

1. In the Enterprise Explorer, select the component and right-click to display its context menu.
2. Click **Make Latest** to display a confirmation dialog box.
   A. If the latest checked-in version of the component has not changed since you retrieved the previous version, you will see the dialog box shown in Figure 50.

Figure 51 Make Latest Dialog Box



   Type in a description of the changes in this version and click **OK**. The version in your workspace will be checked in as the latest version of the component.

   B. If the latest checked-in version of the component is different from the one that was current when you retrieved the previous version, you will first see the dialog box shown in Figure 52.

Figure 52 Confirm Latest Version Override Dialog Box

If you are sure you want to replace the current latest version, click **OK** to display the dialog box shown previously (Figure 51). Type in a description of the changes in this version and click **OK**. The version in your workspace will be checked in as the latest version of the component.

Important: *This situation can occur if another user has made changes to the latest version—you must use caution when checking in your version, since the other user's changes will be superseded.*

Merging Changes from Another Version

A given version of a Collaboration Definition can be updated to a different version of the same Collaboration Definition by using the *Automerge* feature. Two versions of any Collaboration Definition can be saved to a difference (**.sdf**) file, which is an archive file containing the source code for the two versions. This file can be merged with a copy of the *Previous Version* of a given Collaboration Definition to produce a copy of the *Current Version* (refer to Figure 54).

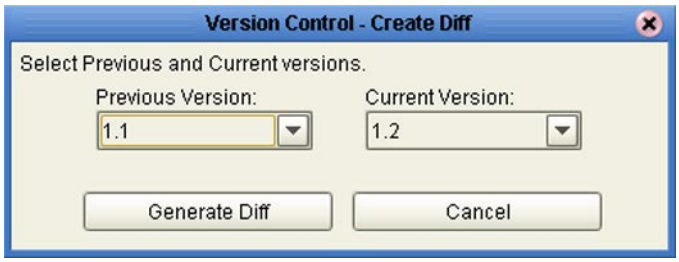
To create a Difference (Diff) file

1 In Project Explorer, right-click the desired Collaboration Definition to display its context menu (see Figure 53).

Figure 53 Collaboration Definition Context Menu

2 Select **Create Diff** to display the dialog box shown in Figure 54.

Figure 54 Version Control - Create Diff Dialog Box



3 Select the two versions of the Collaboration Definition that you want to differentiate.

4 Click **Generate Diff** to display the *Specify Name…* dialog box.

5 Enter a name for the difference file in the *File Name* box and select a directory in which to save the file (or use the defaults).

6 Click **Save** to save the file.

Note: *You can open the* ***.sdf*** *file with a program such as WinZip to examine its contents, which consists of text files containing the Java code of the two versions. The code for the "Previous Version" is saved as* ***source****, and the code for the "Current Version" is saved as* ***modified****. Changes made to the* ***source*** *code are reflected in the* ***modified*** *code.*

To merge a Difference (Diff) file into a Collaboration Definition

1. In Project Explorer, right-click the desired Collaboration Definition to display its context menu.
2. Select **Check Out**.
3. Right-click to again display the context menu.
4. Select **Merge Diff…** to display the dialog box shown in Figure 55.

Figure 55 Version Control - Merge Changes Dialog Box

5. Click the **Ellipsis** (**…**) button to display the *Specify Name…* dialog box.
6. Locate and select the difference (**.sdf**) file you want to merge into the selected Collaboration Definition.
7. Click **Open** add the file to the dialog box shown in Figure 55.
8. Click **Merge** to execute the merger.
9. Check the resulting code and resolve any conflicts resulting from the merger. The differences between the *source* and *modified* code appear between markers shown as <<<<< and >>>>>. You must choose which code to keep, and remove the markers.
10. **Commit** and **Save** the merged Collaboration Definition.

Important: *If you are merging code that contains references to third-party classes or methods, you must import these files into the destination Project before merging the code.*

Command-line Utilities

If you encounter problems with the version control system, there are two command-line utilities—a Repository version control utility and a workspace cleanup script—that can be run by personnel with Administrator privileges. These utilities should be used as a last resort, and with the utmost caution. See the *eGate Integrator System Administration Guide* for information.

Chapter 5

ICAN Projects

This chapter describes components of an ICAN Project, and the use of the Enterprise Designer in defining your Project.

What's in This Chapter

5.1 Overview

An ICAN Project represents the logical system designed to solve either all or part of a business problem. Projects are created using tools contained within the Enterprise Designer, and are deployed to specific Logical Hosts in specific Environments by means of Deployment Profiles (see **Environments** on page 344). Components developed for use in one Project can be used in another, and a Project can internally reference another Project.

Important: *Because of a Java limitation, Project activation can fail on Windows if the path is too long. SeeBeyond recommends that you keep your Projects paths relatively short.*

5.1.1 Project Components

The components found in a typical Project are described in the following sections of this chapter:

- **Services** on page 109
- **External Applications** on page 110
- **Component Connections** on page 113
- **Message Destinations** on page 110

Behind the scenes, and not explicitly shown in a Connectivity Map, are other Project components such as:

- **Collaboration Definitions**

  A Collaboration Definition defines the logical operation taking place in the related Collaboration. It is created in either the Java Collaboration Editor or the XSLT Collaboration Editor, and is based on an Object Type Definition. See **Collaboration Definitions (Java)** on page 184 and **Collaboration Definitions (XSLT)** on page 311.

- **Object Type Definitions**

  Object Type Definitions (OTDs) are sets of rules that define the encoding of an object. They describe messages that are propagated through eGate Integrator, and the methods available for operating on them, and also interactions with external APIs. See **Object Type Definitions** on page 119.

5.2 Project Explorer

A Project consists of logical constructs and configurations designed to solve some or all of a business problem. The **Project Explorer** displays the contents of the Repository that belong to the selected Project (see Figure 56).

Figure 56 Project Explorer

The Project Explorer is used in conjunction with the various editors to create and configure the components of a Project. Each component in the Project Explorer has an icon to identify the component type (see **Project Explorer Icons** on page 94). Right-clicking on a component displays a context menu for that component (see **Context Menus** on page 95), from which you can select appropriate actions.

Note: *Select **Refresh All from Repository** before you open any Project component (such as a Collaboration) to ensure that you open the latest version of the component.*

5.2.1 Project Explorer Icons

The icons described in Table 15 appear in the Project Explorer.

Table 15 Project Icons

Icon	Description
	Represents the **Repository**, which is the central ICAN database where all Project information is saved. Binary files required at run time are also stored here.
	Represents a **Project** or subproject.
	Represents the SeeBeyond **Default Project** or subproject.
	Represents a **Connectivity Map**, which contains the business logic and information about the data transmission. A lock displayed in the lower-left corner indicates that the Connectivity Map is currently checked into the version control system (see OTD example).
	Represents a **Project variable** or **constant**. A lock displayed in the lower-left corner indicates that the variable or constant is currently checked into the version control system (see OTD example).
 	Represents an **Object Type Definition** (OTD) file. A lock displayed in the lower-left corner indicates that the OTD is currently checked into the version control system.
	Represents a **Collaboration Definition (Java)** file. A lock displayed in the lower-left corner indicates that the Collaboration Definition is currently checked into the version control system (see OTD example).
	Represents an **Collaboration Definition (XSLT)** file. A lock displayed in the lower-left corner indicates that the Collaboration Definition is currently checked into the version control system (see OTD example).
	Represents a **Deployment Profile**, which specifies how Project components are deployed to a run-time Environment. A lock displayed in the lower-left corner indicates that the Deployment Profile is currently checked into the version control system (see OTD example).
	Displayed along side one of the above icons, indicates that the current latest version of the component has been checked out for editing.
	Displayed along side one of the above icons, indicates that some version of the component—either the latest version or a previous version—has been retrieved to the local workspace.

5.2.2 Context Menus

Right-clicking a component in the Project Explorer displays a context menu for that component. Only those menu options that are allowed for the component in its current state are activated.

Repository Menu

Figure 57 Repository Menu

Table 16 Repository Menu Options

Option	Function
Project	Adds a new Project to the Repository (see *Note* below).
Sort by Type	Places all objects in order by grouping object types.
Sort by Name	Places all objects in alphabetical order.
Sort by Date	Places all objects in order by creation date, from oldest to newest.
Import	Displays a dialog box with which you can import a Project or Environment into the Repository.
Export	Displays a dialog box with which you can export a Project or Environment from the Repository to an archive file.
Refresh All from Repository	Refreshes the Project Explorer to display the current contents of the Repository. (Open editors are not refreshed.)
User Management	Displays the User Management dialog box, where an Administrator can manage user access to the Repository with options for adding, modifying, and deleting users. See the *eGate Integrator System Administration Guide*.
Properties	Displays a dialog box showing the configuration properties of your Repository.

Note: *Project names should not include special characters such as a comma (,), single quote (‘), or double quote (“). If any of these characters are used, you must escape them when using the import/exportProject command-line tool.*

Project Menu

Figure 58 Project Menu

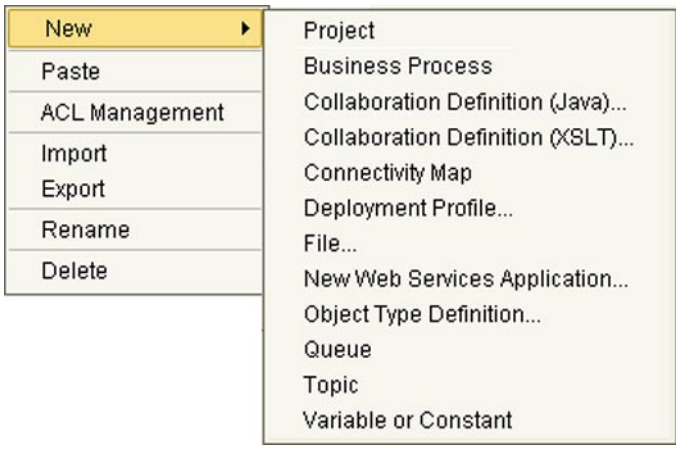


Table 17 Project Menu Options

Option		Function
New	Project	Adds a Subproject folder to the selected Project.
	Business Process	Displays the user interface for creating a new Business Process. (Shown only if eInsight Business Process Manager is installed.)
	Collaboration Definition (Java)	Displays the Collaboration Definition Wizard (Java), with which you can create a Java-based Collaboration Definition. See**Using the Collaboration Definition Wizard (Java)** on page 185.
	Collaboration Definition (XSLT)	Displays the Collaboration Definition Wizard (XSLT), with which you can create an XSLT-based Collaboration Definition. See**Using the Collaboration Definition Wizard (XSLT)** on page 312.
	Connectivity Map	Adds a Connectivity Map to the Project. See **Using the Connectivity Map Editor** on page 107.
	Deployment Profile	Displays a dialog box with which you can create a Deployment Profile for the selected Project. See **Deployment Editor** on page 396.
	File	If the File eWay is installed, displays a dialog box with which you can create an external file to use with the Project (see **File Menu** on page 102).
	New Web Services Appl.	Adds a Web Services Application to the selected Project. See **Web Services Application** on page 417.
	Object Type Definition	Displays the OTD Wizard, with which you can create an Object Type Definition (OTD) file. See **OTD Wizards** on page 123 for more information.
	Queue	Adds a queue to your Project.

Table 17 Project Menu Options

Option		Function
New (continued)	Topic	Adds a topic to your Project.
	Variable or Constant	Displays a dialog box with which you can add a constant or variable icon to your Project.
Paste		Pastes a Collaboration that has been cut or copied from another Project into the selected Project. Only one paste operation can be performed—if you want to paste multiple times, you must recopy the Collaboration each time.
ACL Management		Displays the ACL Properties dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Project. See the *eGate Integrator System Administration Guide*.
Import		Displays a dialog box with which you can import a Project as a Subproject under the selected Project. See **Project/Environment Import** on page 68.
Export		Displays a dialog box with which you can export the selected Project. See **Project/Environment Export** on page 72.
Rename		Activates the field, allowing you to rename the selected Project.
Delete		Deletes the selected Project, subject to the following conditions: ▪ You have *delete* privileges for the Project (see *ACL Management*, above). ▪ The Project is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected Project. Clicking **Yes** then deletes the Project. If the selected Project contains active Deployment Profiles, a dialog box is displayed listing those deployments and requesting that you deactivate them. You must select deactivation to enable the **OK** button.

Note: *Subproject names should not include special characters such as a comma (,), single quote (‘), or double quote (“). If any of these characters are used, you must escape them when using the import/exportProject command-line tool.*

Collaboration Definition Menu

Figure 59 Collaboration Definition Menu

Table 18 Collaboration Definition Menu Options

Command	Function
Open	Opens the appropriate Collaboration Editor, showing the selected Collaboration Definition. See **Using the Collaboration Editor (Java)** on page 193 and **Using the Collaboration Editor (XSLT)** on page 318.
Cut	Copies the selected Collaboration Definition (Java only) and removes it from the current Project, after which you can paste it to another Project (once only). All changes must be committed before you can cut the Collaboration. Cut and paste is disabled for other users when you have the Collaboration checked out.
Copy	Copies the selected Collaboration Definition (Java only), after which you can paste it to other Projects. All changes must be committed before you can copy the Collaboration. You can copy and paste a Collaboration even when another user has the Collaboration checked out.
Version History	Displays a dialog box with which you can track the version history for the selected Collaboration Definition. See **Viewing a Component's Version History** on page 80 for more information.
Check In	Displays a dialog box, with which you can check in a new version of the selected Collaboration Definition. Refer to **Checking a Component In** on page 81 for more details.
Check Out	Displays a dialog box with which you can check out the current version of the selected Collaboration Definition. See **Checking a Component Out** on page 83 for more information.
Undo Check Out	Displays a dialog box with which you can undo the check-out of the selected Collaboration Definition. See **Checking a Component In Without Revisions** on page 82 for more information.

Table 18 Collaboration Definition Menu Options

Command	Function
Make Latest	Allows you to check in the version of the selected Collaboration Definition that was retrieved to your workspace, making it the latest version. See **Checking In a Previous Version as the Latest Version** on page 86.
Create Diff	Displays a dialog box with which you can create a difference file representing two versions of the selected Collaboration Definition. See **Merging Changes from Another Version** on page 88.
Merge Diff	Displays a dialog box with which you can merge modifications into a specific version of the selected Collaboration Definition. See **Merging Changes from Another Version** on page 88.
Delete	Deletes the selected Collaboration Definition, subject to the following conditions: ▪ You have *delete* privileges for the Collaboration Definition (see *ACL Management*, below). ▪ The Collaboration Definition is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected Collaboration Definition. Clicking **Yes** then deletes the Collaboration Definition.
ACL Management	Displays the ACL Properties dialog box, with which an Administrator can assign read/write privileges to users for the selected Collaboration Definition. See the *eGate Integrator System Administration Guide*.
Properties	Displays the appropriate Collaboration Definition Properties dialog box for the selected Collaboration Definition.

Note: *The version history for a component that has been **cut** and pasted is preserved, since there can be only one instance of it. The version history for a component that has been **copied** and pasted is **not** preserved, since there can be multiple instances of it; the version number for each pasted instance is reset.*

Note: *If a component is copied and pasted back into the original Project, the name is automatically modified with a suffix (_1); in the case of multiple pastes, the suffix is incremented by 1 for each subsequent paste.*

Connectivity Map Menu

Figure 60 Connectivity Map Menu

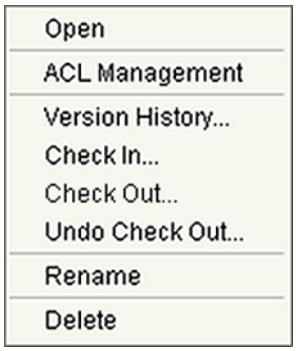


Table 19 Connectivity Map Menu Options

Command	Function
Open	Opens the Connectivity Map Editor, showing the selected Connectivity Map. See **Using the Connectivity Map Editor** on page 107.
ACL Management	Displays the ACL Properties dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Connectivity Map. See the *eGate Integrator System Administration Guide*.
Version History	Displays a dialog box with which you can track the version history for the selected Connectivity Map. See **Viewing a Component's Version History** on page 80 for more information.
Check In	Displays a dialog box, with which you can check in a new version of the selected Connectivity Map. See **Checking a Component In** on page 81 for more details.
Check Out	Displays a dialog box with which you can check out the current version of the selected Connectivity Map. See **Checking a Component Out** on page 83 for more information.
Undo Check Out	Displays a dialog box with which you can undo the check-out of the selected Connectivity Map. See **Checking a Component In Without Revisions** on page 82 for more information.
Rename	Activates the field, allowing you to rename the selected Connectivity Map.
Delete	Deletes the selected Connectivity Map, subject to the following conditions: ▪ You have *delete* privileges for the Connectivity Map (see *ACL Management*, above). ▪ The Connectivity Map is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected Connectivity Map. Clicking **Yes** then deletes the Connectivity Map.

Deployment Profile

Figure 61 Deployment Profile Menu

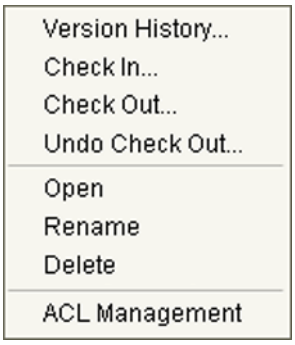


Table 20 Deployment Profile Menu Options

Command	Function
Version History	Displays a dialog box with which you can track the version history for the selected Deployment Profile. See **Viewing a Component's Version History** on page 80 for more information.
Check In	Displays a dialog box, with which you can check in a new version of the selected Deployment Profile. See **Checking a Component In** on page 81 for more details.
Check Out	Displays a dialog box with which you can check out the current version of the selected Deployment Profile. See **Checking a Component Out** on page 83 for more information.
Undo Check Out	Displays a dialog box with which you can undo the check-out of the selected Deployment Profile. See **Checking a Component In Without Revisions** on page 82 for more information.
Open	Opens the Deployment Editor, showing the selected Deployment Profile. See **Deployment Editor** on page 396.
Rename	Activates the field, allowing you to rename the selected Deployment Profile.
Delete	Deletes the selected Deployment Profile, subject to the following conditions: ▪ You have *delete* privileges for the Deployment Profile (see *ACL Management*, above). ▪ The Deployment Profile is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected Deployment Profile. Clicking **Yes** then deletes the Deployment Profile.
ACL Management	Displays the ACL Properties dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Deployment Profile. See the *eGate Integrator System Administration Guide*.

File Menu

Figure 62 File Menu

Table 21 File Menu Options

Command	Function
Version History	Displays a dialog box with which you can track the version history for the selected file. See **Viewing a Component's Version History** on page 80 for more information.
Check In	Displays a dialog box, with which you can check in a new version of the selected file. See **Checking a Component In** on page 81 for more details.
Check Out	Displays a dialog box with which you can check out the current version of the selected file. See **Checking a Component Out** on page 83 for more information.
Update	Displays a file browser dialog box with which you can replace the file with an updated version.
Export	Displays a file browser dialog box with which you can save the selected file to another directory.
Delete	Deletes the selected file, subject to the following conditions: ▪ You have *delete* privileges for the file (see *ACL Management*, above). ▪ The file is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected file. Clicking **Yes** then deletes the file.

Object Type Definition Menu

Figure 63 OTD Menu

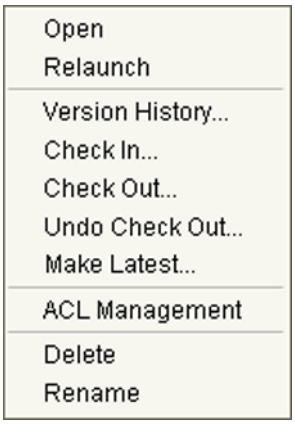


Table 22 OTD Menu Options

Command	Function
Open	Opens the OTD Editor, showing the selected Object Type Definition. See **OTD Editor** on page 125.
Relaunch	Relaunches the appropriate OTD Wizard, so that you can re-define the selected Object Type Definition while retaining the original OID. The following conditions apply: ▪ You must have *write* privileges for the OTD (see *ACL Management*, above). ▪ You must select the same top level node when relaunching the OTD. ▪ The OTD must not be checked out by anyone other than yourself. ▪ The OTD must not be imported from another Project. ▪ The OTD must have been created in eGate Integrator 5.0.4 (or later release). ***Note:*** *Does not apply to User-Defined OTDs.*
Version History	Displays a dialog box with which you can track the version history for the selected Object Type Definition. See **Viewing a Component's Version History** on page 80 for more information.
Check In	Displays a dialog box, with which you can check in a new version of the selected Object Type Definition. See **Checking a Component In** on page 81 for more details.
Check Out	Displays a dialog box with which you can check out the current version of the selected Object Type Definition. See **Checking a Component Out** on page 83 for more information.
Undo Check Out	Displays a dialog box with which you can undo the check-out of the selected Object Type Definition. See **Checking a Component In Without Revisions** on page 82 for more information. ***Note:*** *Does not apply to WSDL OTDs.*

Table 22 OTD Menu Options

Command	Function
Make Latest	Allows you to check in the version of the selected Object Type Definition that was retrieved to your workspace, making it the latest version. See **Checking In a Previous Version as the Latest Version** on page 86. ***Note:*** *Does not apply to WSDL OTDs.*
ACL Management	Displays the ACL Properties dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Object Type Definition. See the *eGate Integrator System Administration Guide*.
Delete	Deletes the selected Object Type Definition, subject to the following conditions: ▪ You have *delete* privileges for the OTD (see *ACL Management*, above). ▪ The OTD is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected OTD. Clicking **Yes** then deletes the OTD.
Rename	Activates the field, allowing you to rename the selected Object Type Definition.

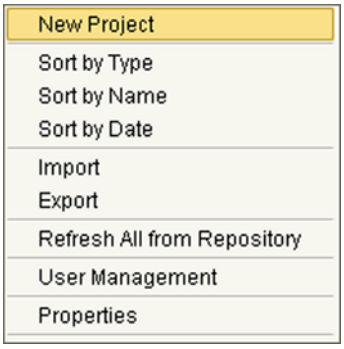
5.3 Creating a Project

You can create an ICAN Project by using the context menu system in Enterprise Explorer together with the Connectivity Map Editor. This section contains a brief outline of the procedure to get you started—an end-to-end description can be found in the *eGate Integrator Tutorial*.

To create an ICAN Project

1 Right-click the **Repository** in the Project Explorer to display its context menu (see Figure 64).

Figure 64 Repository Context Menu - New Project

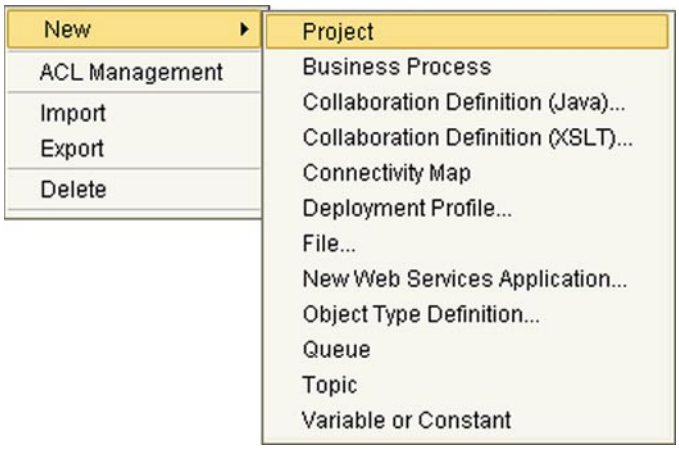


2 Select a **New Project**, which initiates a hierarchical structure under the Repository.

3 Rename the Project as desired.

4 Right-click the **Project** to display its context menu (see Figure 65).

Figure 65 Project Context Menu - New Project



5 Select a **New Connectivity Map**, which displays the Connectivity Map Editor.

6 In the Connectivity Map Editor, populate the canvas with the various components needed for your Project (see **Using the Connectivity Map Editor** on page 107), but leave the connectivity for later.

7 After you have developed the basic architecture of your Project, return to the Project context menu in Project Explorer.

8 Select a **New Business Process**, if applicable (you must have eInsight Business Process Manager installed).

9 Select a **New Object Type Definition**, and build your OTD as described in **Object Type Definitions** on page 119.

10 Select a **New Collaboration Definition (Java or XSLT)**, and build your Collaboration Definition(s) as described either in **Collaboration Definitions (Java)** on page 184 or **Collaboration Definitions (XSLT)** on page 311.

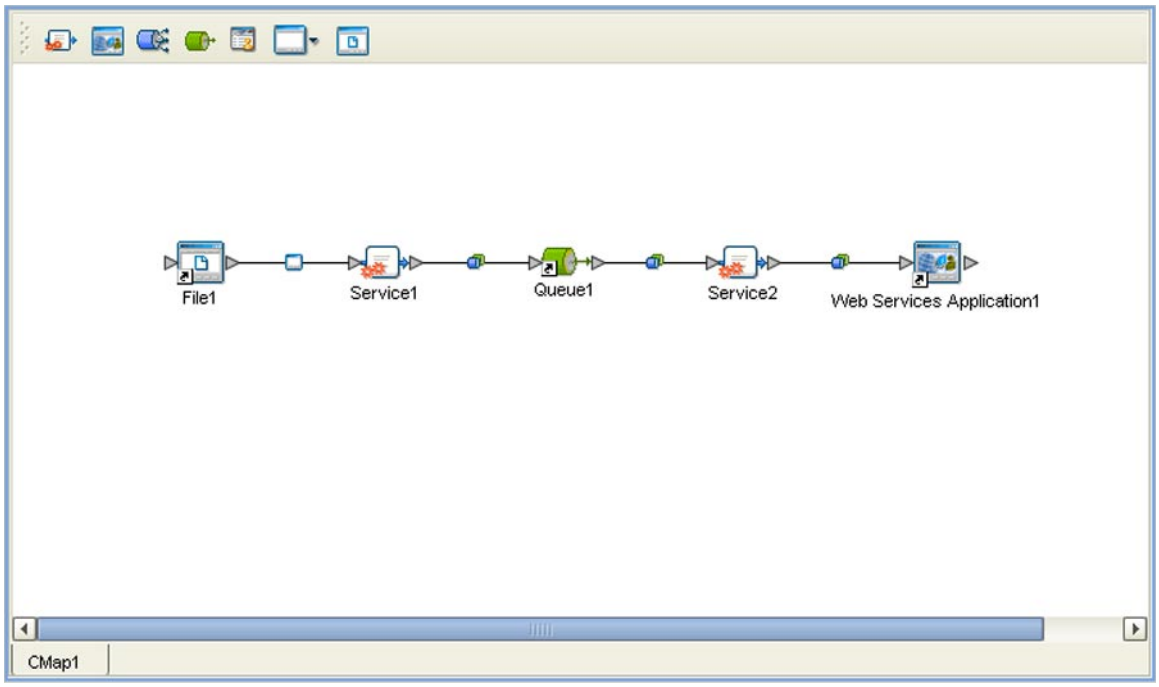
11 Return to the Connectivity Map and connect the various components (see **Using the Connectivity Map Editor** on page 107).

12 Save and test your Project.

5.4 Using the Connectivity Map Editor

When you create a new Connectivity Map in the Enterprise Explorer, the editor panel displays the Connectivity Map Editor (see Figure 66). To define your Project, you simply drag icons from the toolbar to the workspace, or canvas, to populate the Connectivity Map with the necessary components. You subsequently link the components by dragging the cursor from one to the other.

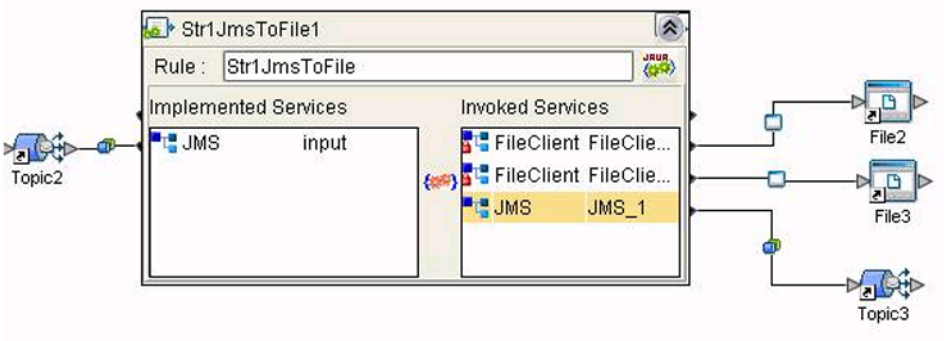
Figure 66 Connectivity Map Editor



The drag-and-drop components include services, queues, topics, schedulers, and external applications. Additional components, such as eWays and JMS Clients, are placed automatically when you link the components you have placed manually.

When there are multiple destinations, as with a JMS topic, the Connectivity Map Editor cannot resolve which output port connects to which destination. Because of this, the Collaboration definition must be created first, and the connections must be drawn by opening the Collaboration Binding box in Connectivity Map (see Figure 67).

Figure 67 Linking JMS Topics



The Connectivity Map Editor toolbar contains the icons listed in Table 23, plus additional icons representing eGate Integrator add-ons and other ICAN components that you may have installed.

Table 23 Connectivity Map Toolbar Icons

Icon	Component	Function
	Service	A logical component that provides the framework for a process or Collaboration. See **Service Component** on page 109.
	Queue	A Message Destination that conforms to the point-to-point messaging paradigm, having one sender and one receiver. See the *eGate Integrator JMS Reference Guide* for information.
	Topic	A Message Destination that conforms to the publish/subscribe messaging paradigm, having one sender (publisher) and multiple receivers (subscribers). See the *eGate Integrator JMS Reference Guide for* information.
	Web Services Application	Represents a Web Services Application (see **Web Services Application** on page 417).
	External Applications	Represents an application external to eGate Integrator. Click the arrow beside the icon to view a list of specific applications to which you can connect. See **External Application Drop-Down Menu** on page 110.
	Scheduler	Represents a scheduling component of the Connectivity Map. Use this component to set data transfer to occur at set intervals. See **Schedulers** on page 111.

It is important to understand that the logical components appearing in the Connectivity Map are essentially *placeholders* that refer to the "actual" components that exist in the Repository and appear in the Project Explorer. Renaming or deleting a queue or topic in the Connectivity Map only affects the placeholder, not the object in the Repository.

Also, renaming or deleting a queue or topic in the Repository will not affect the existence or name of the associated placeholder in the Connectivity Map. The change will, however, be reflected in the *tooltips* for the placeholder. This allows you to re-assign the placeholder without disrupting the continuity of the Connectivity Map.

5.5 Services

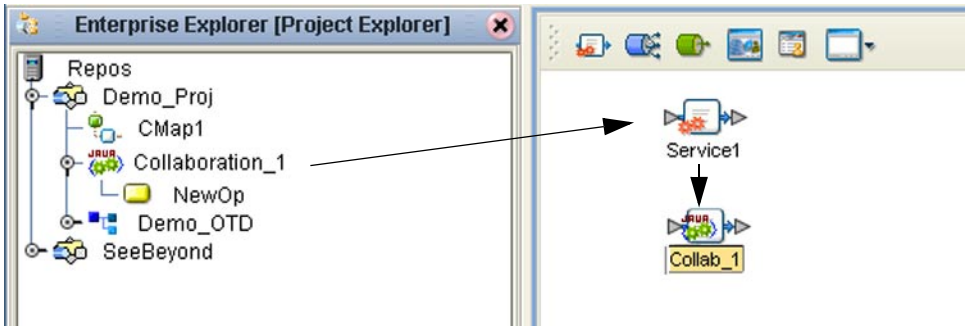
A service provides a framework for a process or a Collaboration, which contains the information required to execute a set of business rules.

5.5.1 Collaborations

A Collaboration is a logical operation performed between some combination of message destinations and external applications. The operation is defined by a Collaboration Definition, which can be encoded in either Java or XSLT.

The Collaboration acts as a service having a publication or subscription relationship with each linked entity. The link is provided by a JMS Client connection (see **Component Connections** on page 113. Dragging a Collaboration from the Project Explorer to the Service icon in the Connectivity Map defines the service as a Collaboration (see Figure 68).

Figure 68 Service Component



Note: *Any changes made to the names of the Collaborations should be done when the Collaboration is created. If the name is changed later, the Collaboration should be opened, regenerated if applicable, and saved again. For safe measure, this should also be done before creating the Connectivity Map and Deployment Profile.*

Connection-related properties for the Collaboration (or other service) are configured in the adjoining JMS Client. These properties include:

- Concurrent or serial processing
- Transaction mode (transacted or XA)
- Security

All properties, and the procedures for configuring them, are detailed in the *eGate Integrator JMS Reference Guide*.

5.6 Message Destinations

A message destination is a container for stored data, and can follow either the topic or queue JMS model.

- A *topic* is a message destination that conforms to the publish-and-subscribe messaging paradigm.
- A *queue* is a message destination that conforms to the point-to-point messaging paradigm.

See the *eGate Integrator JMS Reference Guide* for information on message destinations.

5.7 External Applications

The basic purpose of eGate Integrator is to facilitate the interchange of data between external business applications. These business applications are collectively referred to as external applications, and are represented in the Project by logical proxies for the specific applications involved. An external application can be identified with an ERP application such as SAP or PeopleSoft, a DBMS such as Oracle or SQL, or with a particular communications protocol, such as TCP/IP or HTTPS.

External applications are logical representations of external software applications that are being integrated by the eGate Integrator system. These are linked to a Service by means of an eWay. Clicking the drop-down arrow beside the external application icon displays a menu showing those applications corresponding to eWays that have been purchased and installed, plus the Scheduler. An example is shown in Figure 69.

Figure 69 External Application Drop-Down Menu

Selecting the check box beside an individual external application adds that icon to the toolbar; clearing the check box removes it from the toolbar.

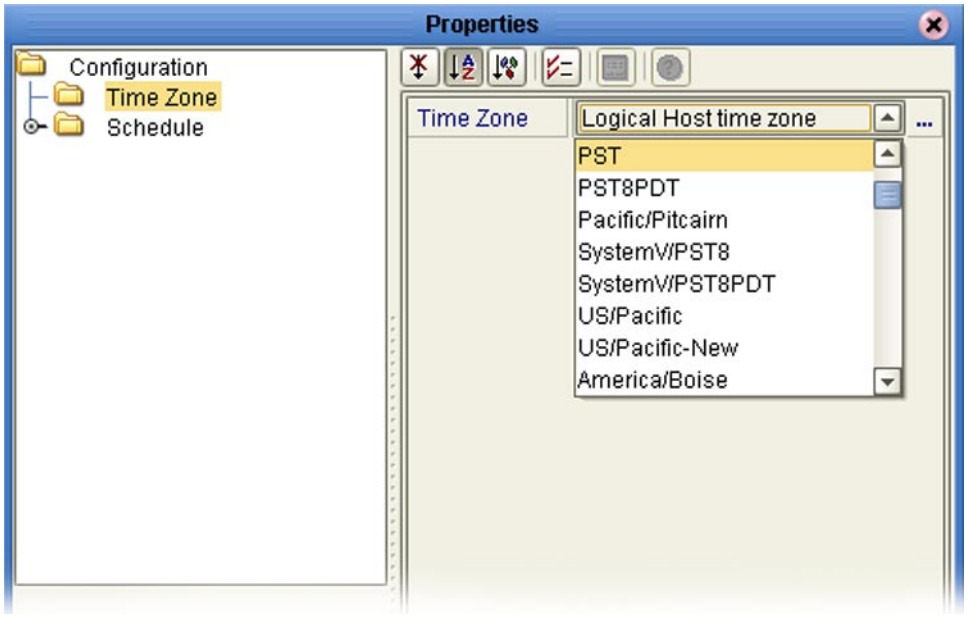
5.7.1 Schedulers

A Scheduler allows a service to be performed at a prescribed interval. Once the scheduler is connected to a service in the Connectivity Map, double-clicking the JMS Client (the connector icon) displays the *Properties* dialog box for that scheduler. Double-clicking the properties field enables it for editing.

Time Zone

Selecting **Time Zone** displays the *Time Zone* property field in which you specify your local time zone, so that your schedule will be synchronized to the local time (see Figure 70).

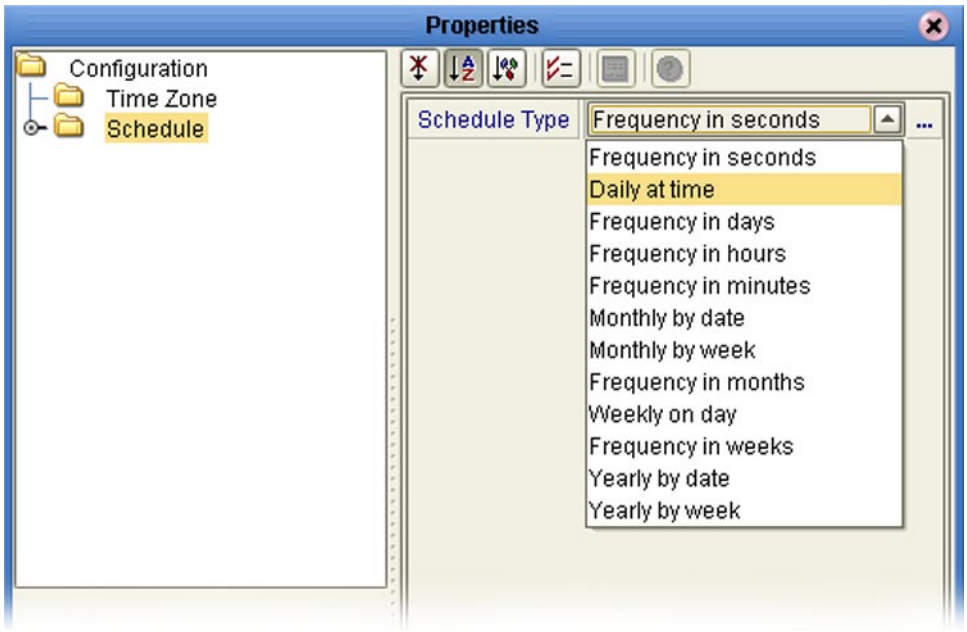
Figure 70 Scheduler Properties Dialog Box - Time Zone



Schedule Type

Selecting **Schedule** displays the *Schedule Type* property field which you set to the type of schedule you want to use (see Figure 71).

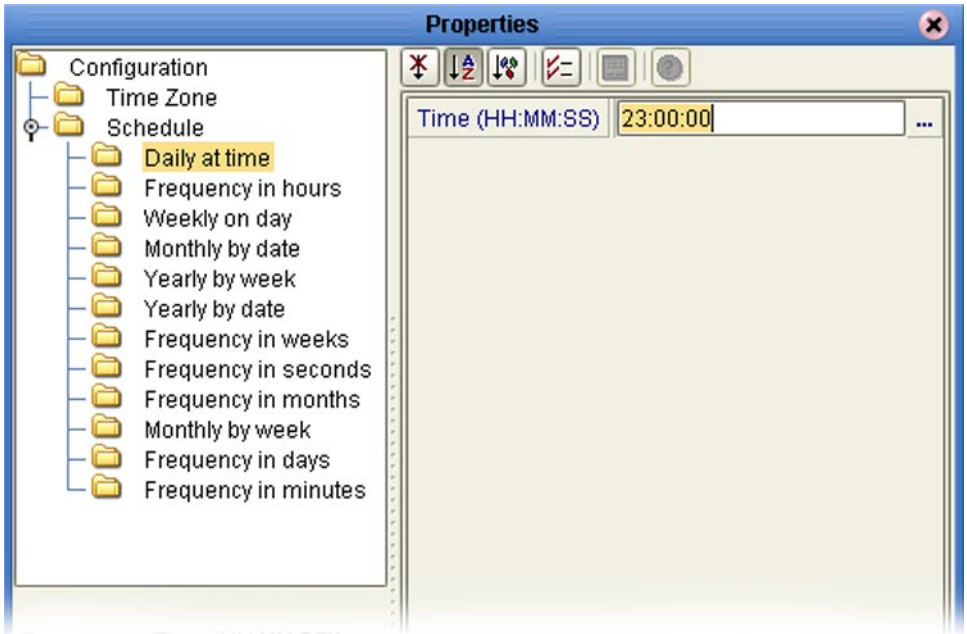
Figure 71 Scheduler Properties Dialog Box - Schedule Type



Schedule Value

Selecting a **Schedule Type** node in the explorer tree displays the property field for that schedule type, in which you specify the desired value (see Figure 72).

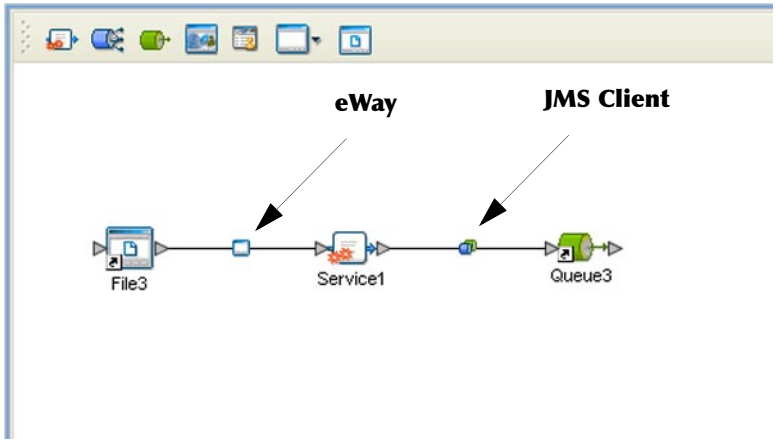
Figure 72 Scheduler Properties Dialog Box - Schedule Value



5.8 Component Connections

When you link two components on a Connectivity Map, the Enterprise Designer places either an eWay or JMS Client connection icon on the link, depending upon the type of components you are linking (see Figure 73).

Figure 73 Connection Icons in a Connectivity Map

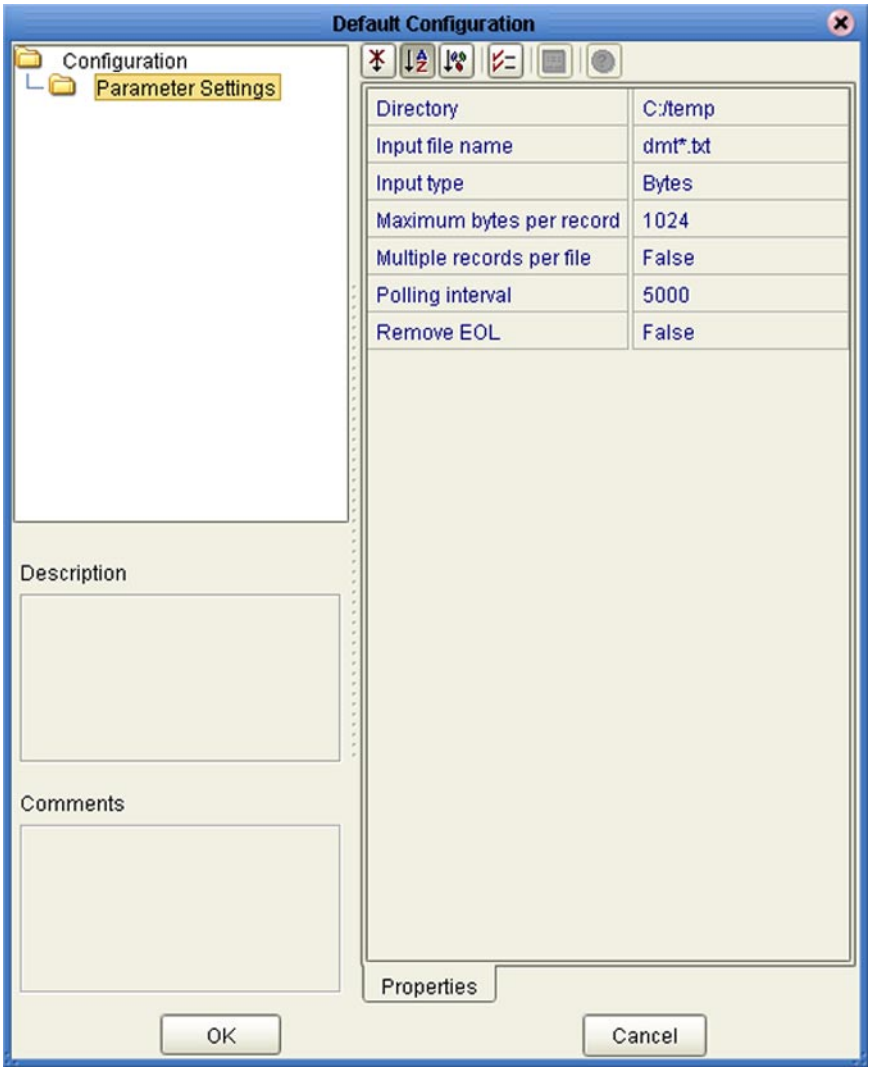


- When you link an external application with a Collaboration, the Enterprise Designer automatically adds an eWay Connection icon to the link. The eWay enables communication and movement of data between the external application and the eGate Integrator system. The eWay configuration specifies the logical connection properties for the link. See the individual eWay Intelligent Adapter User's Guides for specific information.
- When you link a Service with a Message Destination (queue or topic), the Enterprise Designer adds a JMS Client Connection icon. The JMS Client configuration specifies the logical connection properties for the linked Service. See the *eGate Integrator JMS Reference Guide* for information.

5.8.1 Configuring a Connection

Double-clicking an eWay or JMS Client connection icon in the Connectivity Map displays the Default Configuration dialog box. As an example, Figure 74 shows a dialog box that lists the configuration properties for a File eWay.

Figure 74 Default Configuration Dialog Box



Note: *The first time you double-click an eWay or JMS Client icon, you will see a Templates dialog box. Here, you must designate an eWay to be inbound or outbound. Clicking* ***OK*** *will then display the Default Configuration dialog box.*

The constituent parts of the Default Configuration dialog box are:

- The **Configuration Tree** includes folders that contain configuration and connection properties for the selected eWay or message destination.
- The **Toolbar** contains a series of buttons used to sort and modify the information listed in the Properties folder, as described in Table 24.
- The **Description** box contains a brief description of the contents of the item currently selected in the Configuration Tree.
- The **Comments** box is for user comments about the item selected in the Configuration Tree.

Table 24 Configuration Dialog Box Toolbar Buttons

Button	Command	Function
	Unsorted	Displays configuration properties in their default order.
	Sort by Name	Sorts configuration properties alphabetically by name.
	Sort by Type	Displays configuration properties by property type.
	Show Editable Properties Only	Displays only the properties of an eWay or message destination that can be modified.
	Customizer	Displays the **Customizer** dialog box, which you can use to customize the selected eWay or message destination.
	Help	Displays the online help documentation for the Configuration Editor.

5.9 Defining Constants and Variables

You can define both constants and variables for use in a specific Project.

Project Constants

Constants are name/value pairs that are visible across the Project. For example, Figure 75 shows a standard currency defined to be used globally throughout the system.

Figure 75 Creating a Project Constant

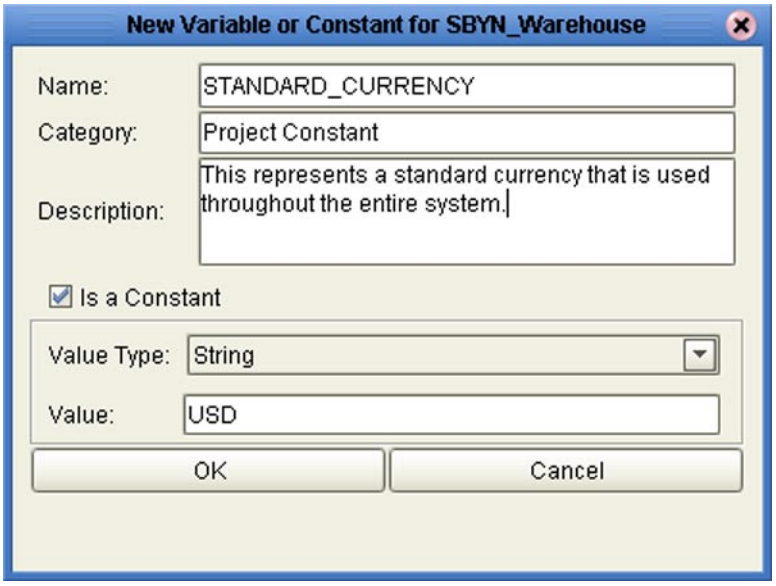


Table 25 Project Variable Options

Property	Option	Description/Usage
Name		Your name for the Project constant.
Category		You may assign a category name, if desired.
Description		Your description for the constant.
Is a Constant		Select to enable the constant properties fields.
Value Type	String	Allows the string value to be displayed explicitly.
	Password	Encrypts the value, displaying asterisks (*) in the field.
Value		The value for the constant.

Project Variables

Variables function as placeholders, having values that are determined when you create a specific Deployment Profile (see **Mapping Variables** on page 401). Project variables can be literals or Environmental constants.

As an example, Figure 76 shows a Project variable defined to represent a password of a database user in a target Environment. A system manager assigns an actual value to this variable in the Deployment Profile editor. The value of the assigned Project variable—an Environmental constant— is then used to connect to the database in the target Environment.

Figure 76 Creating a Project Variable

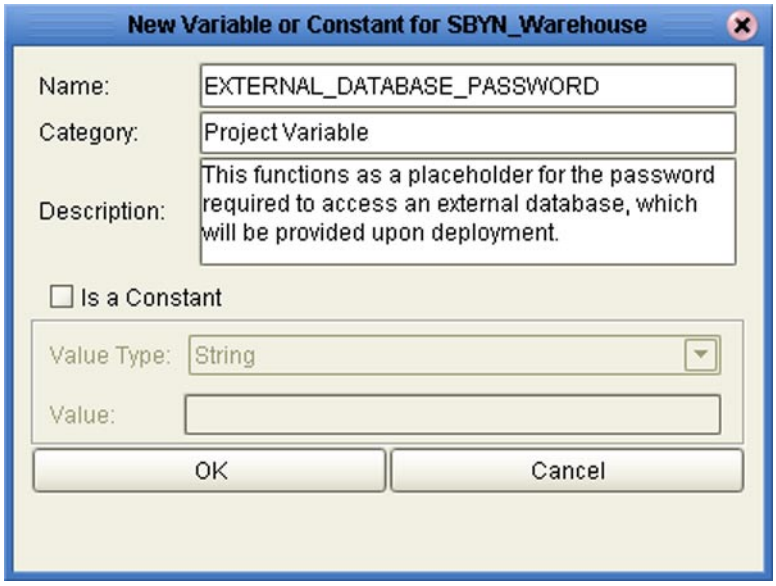


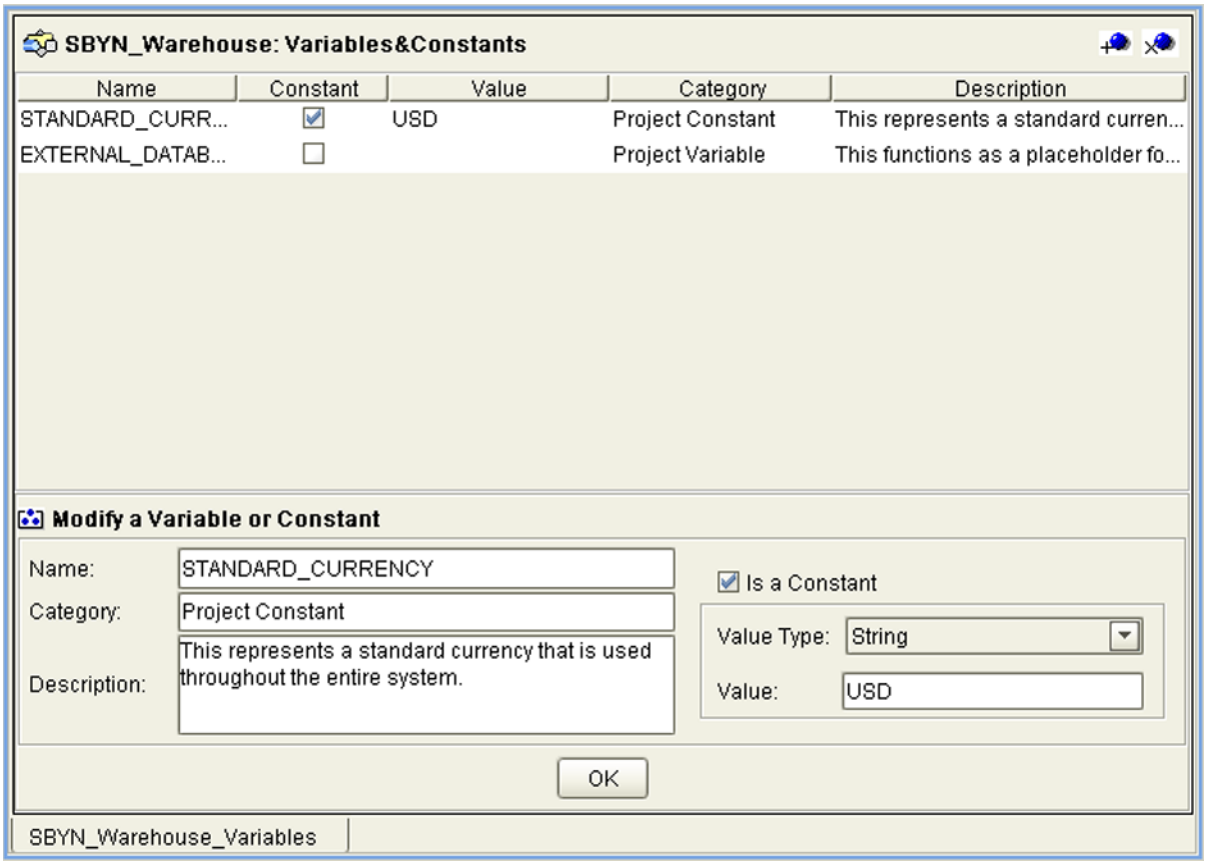
Table 26 Project Variable Options

Property	Description/Usage
Name	Your name for the Project variable.
Category	You may assign a category name, if desired.
Description	Your description for the variable.
Is a Constant	Does not apply to variables—leave unchecked.
Value Type	Does not apply to variables.
Value	Does not apply to variables.

Variables & Constants Object Group

Constants and variables are automatically added to a Variables and Constants object group within the Project (see Figure 77). Selecting an entry displays it in the lower panel, where it can be modified. Clicking **OK** overwrites the previous definition with the modified version.

Figure 77 Variables and Constants Object Group



Chapter 6

Object Type Definitions

This chapter introduces Object Type Definitions (OTDs), describes the features of the OTD Editor, and outlines procedures for creating both externally-defined and user-defined OTDs.

What's in This Chapter

6.1 Overview

The basic functionality of eGate Integrator is handling messages, which it accomplishes by means of Collaborations. To operate on a message, the Collaboration needs a description of the message format. The format description may follow a standard, and be available in some standard metadata format, or it may not be; in the latter case, you need a convenient way to define the format. OTDs provide the solution.

OTDs describe external data formats that characterize the input and output data structures in a Collaboration Definition. OTDs typically have a specific external representation format that is used to store and transport the OTD contents through an ICAN Project. The OTD defines both this external representation and the run-time data structure.

Typically, a Collaboration will receive a message containing the external representation of a particular OTD. It will use the *unmarshal* method of an instance of that OTD to parse the data and make it accessible though the hierarchical data structure. Then it will perform some operation—for example, copying parts of the data to another OTD instance. Finally, it will invoke the *marshal* method on the other OTD instance to render the contents of its data structure as a single, serialized data stream for further transport.

At run time, an OTD instance is accessed either directly from Java in a Java-based Collaboration, using accessors resembling Java beans, or from BPEL using XPath

expressions. In the case of Java, each of the nodes comprising the hierarchy of the data structure has a set of properties with *get* and *set* methods.

Important: *If you delete an OTD in the Project Explorer, any Collaboration Definitions that have been built using that OTD will be affected. It is recommended that you run the Impact Analyzer before attempting to delete any OTDs (see* **Impact Analyzer** *on page 77).*

As with other eGate Integrator components, it is essential to manage versions of Object Type Definitions carefully. See **Version Control** on page 79 for descriptions of various version control features applicable to Object Type Definitions.

6.1.1 OTD Fundamentals

Here we introduce some basic concepts and terminology. This information applies to OTDs in general, but is of primary interest in regard to user-defined OTDs.

Functionality

An OTD provides three basic sets of information:

- The definition of a hierarchical data structure forming the Java-accessible representation of a message, internal to eGate Integrator.
- The definition of a grammar to scan an input message in its external representation, and rules on how to map the result to the internal representation (an operation known as *unmarshaling* or *parsing*).
- The definition of rules to generate the external representation of an output message from the internal representation (an operation called *marshaling* or *serialization*).

Data Structure

The run-time data structure is composed of a hierarchical system of *nodes*. These nodes are characterized by terms indicating their relationships with each other:

Parent, Child, and Sibling Nodes

Any subnode of a given node is called a *child* node, and the given node, in turn, is the child's *parent*. *Sibling* nodes are nodes on the same hierarchical level under the same parent node. Nodes higher than a given node in the same lineage are *ancestors* and those below it are *descendants*.

Root and Leaf Nodes

The *root* node is the highest node in the tree structure, and has no parent. This node represents the entire OTD. It may have one or more child notes, but can never have sibling nodes or be repeating. The properties of the root node (other than its name) cannot be edited.

Leaf nodes have no children, and normally carry the actual data from the message. *Non-leaf* nodes, which have children, provide the framework through which this data is accessed and organized.

Non-leaf Nodes

There are two basic types of non-leaf nodes:

- *Group* nodes, which provide organizational grouping for purposes such as repetition.
- *Choice* nodes, which represent sets of alternatives—only one of which is valid at any given time for an instance of that node. For example, a choice node named **order** might have two children, respectively named **domestic** and **overseas**. For each order, only one of these children will be present.

  At the bean interface level, a choice node is characterized by having a *hasX* method for each child *X*, and a *choice* method that returns the index of the currently present child (**0** for first child, **1** for second child, and so on). Calling *getX* causes child *X* to be present, creating it if necessary. The *optional* and *repeating* flags cannot be set for the children of a choice node.

Data Types

The basic data types are *fixed* and *delimited*, but there are variations to these (see **Specifying the Node Type** on page 167).

- With fixed-length data, the length of the unit of data is always the same. The position of data is described by *byte offset* and *length*.
- With delimited data, the length of the unit of data is variable. Information is separated by a pre-determined system of delimiters defined within the properties of the OTD (see **Specifying Delimiters** on page 168).

Name Properties

An OTD node has two distinct name properties: a *display name*, which is essentially an arbitrary string, and a *Java name*, which is the accessor basename. (For example, if a node has the Java name *ElementX*, then the implementing class for that node will contain a method *getElementX*.) The Java name is normally derived from the display name, modified to suit the restrictions on Java identifiers, and supplied automatically by eGate Integrator.

Important: *Do not modify the Java name property.*

Note: *To be compatible with BPEL, an OTD cannot support a pure string-BLOB.*

6.1.2 OTD Types

Externally-Defined OTDs

Externally-defined OTDs are based on formats or standards external to eGate Integrator, such as Document Type Definition (DTD), Web Services Definition Language (WSDL), XML Schema Definition (XSD), and various proprietary formats such as SAP BAPI. Some of these OTDs are *messagable*, others are API-based. Externally-defined OTDs are read-only. See **Externally-Defined OTDs** on page 135.

User-Defined OTDs

User-defined OTD are native to eGate Integrator. You can create a User-defined OTD from scratch using the User-Defined OTD Wizard and the OTD Editor. User-defined OTDs are read/write—you can add or delete nodes and edit their properties. See **User-Defined OTDs** on page 153.

JMS OTDs

For Collaborations that read from and write to topics or queues, you must add the JMS OTD to the Collaboration Definition. This indicates to the Collaboration which topic or queue it expects to receive messages from or send messages to, and allows you to build the JMS business rules. JMS OTDs are described in the *eGate Integrator JMS Reference Guide*.

OTD Libraries

Several libraries containing large numbers of OTDs for specific purposes are available to use with eGate Integrator. These OTDs are pre-defined templates corresponding to message types used by SWIFT, HL7, and other industry-specific data exchange systems. See **OTD Libraries** on page 183.

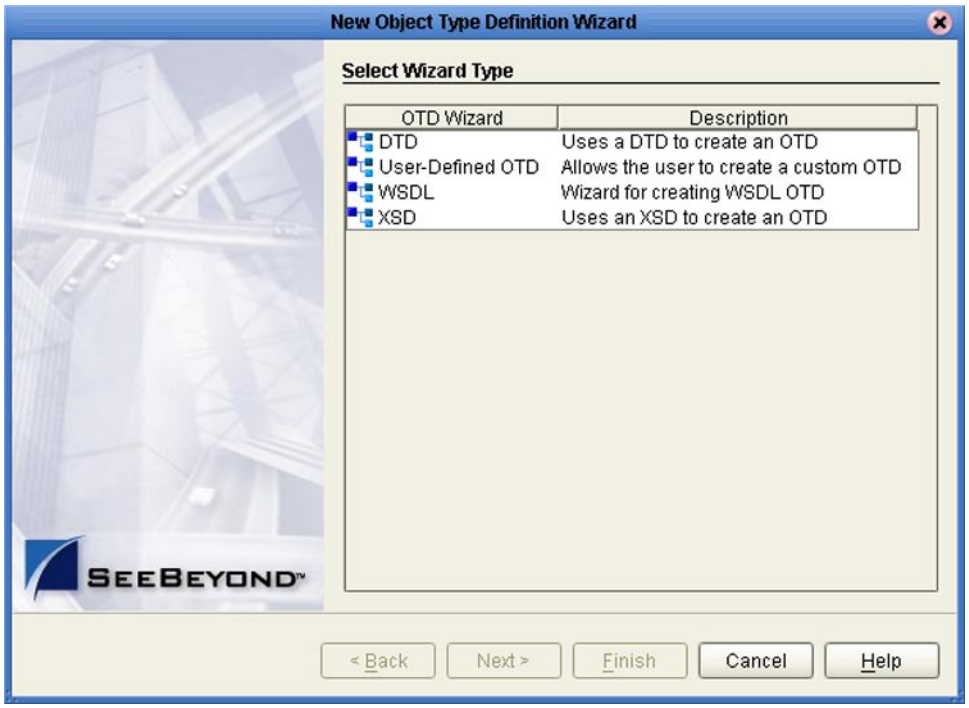
6.2 OTD Wizards

Wizards are provided in the Enterprise Designer GUI to guide you through the OTD building process. These Wizards call back-end builders that actually implement the building of the code, based on the provided information.

Right-click on a Project in the Enterprise Explorer to display the Project context menu, then select **New Object Type Definition** to display the OTD Wizard selection dialog, shown in Figure 78. This initial dialog allows you to select the specific type of OTD Wizard needed for your application. The basic Wizards supplied with eGate Integrator are described in:

- **Using the DTD Wizard** on page 135
- **Using the WSDL Wizard** on page 141
- **Using the XSD Wizard** on page 147
- **Using the User-Defined OTD Wizard** on page 153

Figure 78 OTD Wizard Selection Dialog



Additional OTD Wizards are supplied with eGate Integrator add-on components, and are described in the User's Guides for the specific products. When these products are installed, the OTD Wizards are added to the list shown in Figure 78.

The OTD Wizards guide you through the initial phases of creating an Object Type Definition, and then invoke the OTD Editor. The user interface is highly self-explanatory, but details of the navigation buttons are listed in Table 27 for your reference.

Table 27 OTD Wizard Navigation Buttons

Button	Function
< Back	Returns to the previous step in the wizard. This button is disabled on the first step.
Next >	Goes to the next step in the wizard. This button is disabled on the last step.
Finish	Saves all OTD settings and closes the wizard. This button is only enabled on the last step.
Cancel	Closes the wizard without saving the OTD.
Help	Displays the online help documentation for the **OTD Wizard** dialog box.

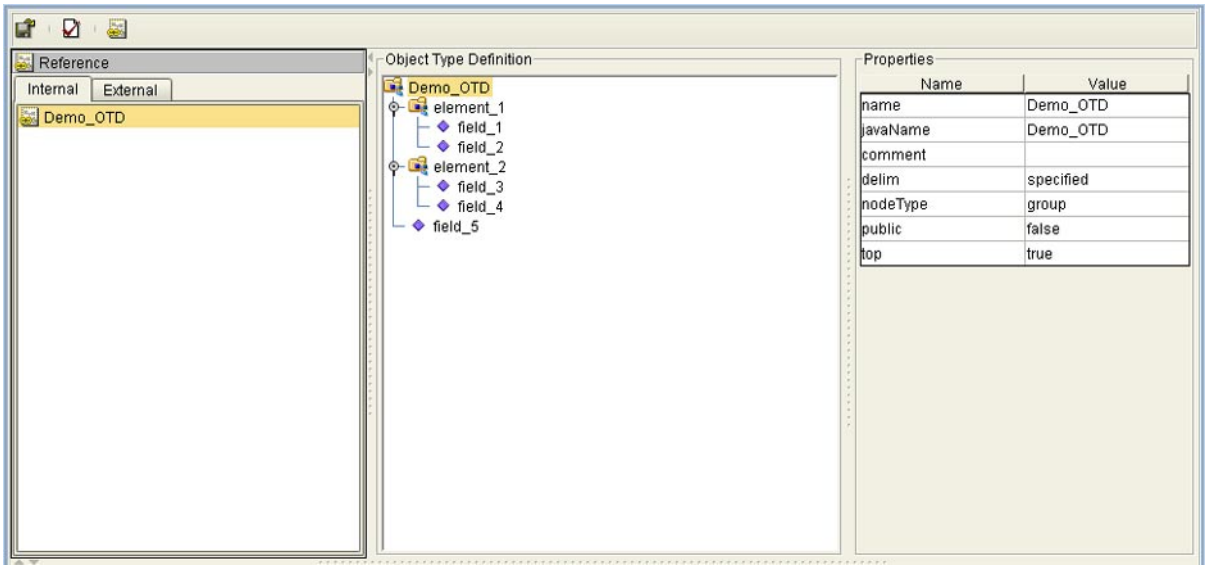
6.3 OTD Editor

After you create an OTD file using the OTD Wizard, the OTD Editor appears in the editor panel of the Enterprise Designer, as shown in Figure 79. You can also invoke the OTD Editor by selecting **Open** in the context menu for an existing OTD in the Project Explorer. OTDs are saved to the Project automatically.

Use of the editing features of the OTD Editor are described in **User-Defined OTDs** on page 153.

Note: *Remember that externally-defined OTDs are read-only, and cannot be edited. You can, however, test them to verify correctness of the build.*

Figure 79 OTD Editor



Major features of the OTD Editor interface are:

- **Reference**
  This area contains internal and external templates for the OTD file.
- **Object Type Definition**
  This area displays each field and element included in the OTD file.
- **Properties**
  This area displays details about the OTD file or field selected in the *Object Type Definition* list.

Table 28 OTD Editor Toolbar Icons

Icon	Command	Function
	Save as New Name to Repository	Displays a dialog box with which you can save the current OTD under a new name to any Project or Subproject in the Repository. 
	Tester	Displays/refreshes the Tester area.
	Toggle Reference Tab Panel	Displays/hides the Reference area.

6.3.1 OTD Tester

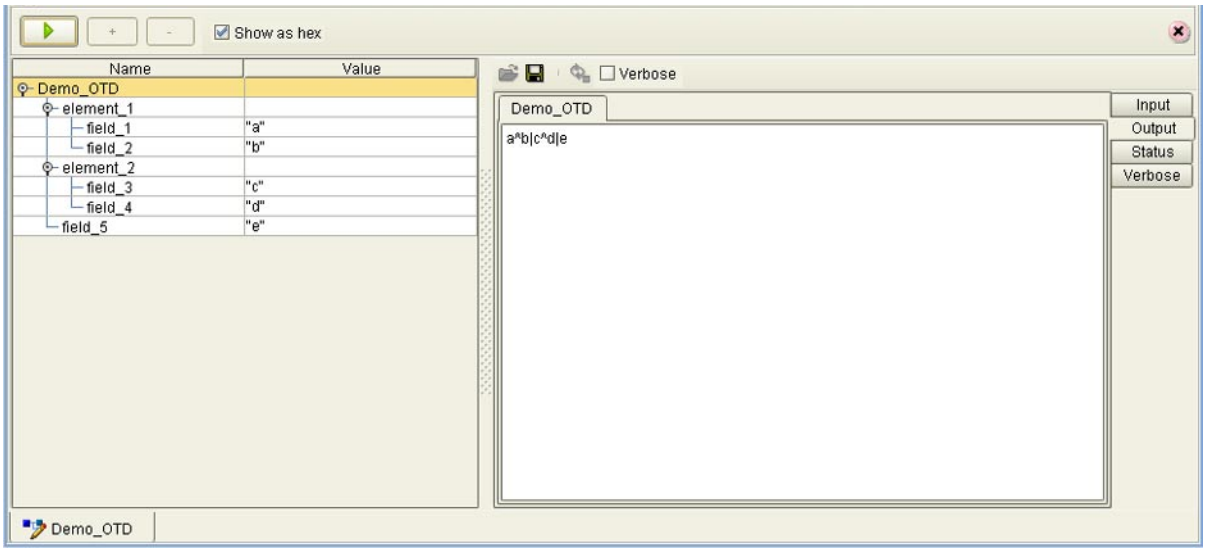
The OTD tester allows you to simulate the operation of a Collaboration on an OTD, thereby checking the correctness of the OTD during the design phase; for example to:

- Prevent data errors at run time.
- Verify that all required data elements are available.
- Verify that all used data formats are correct.

You can enter input data values, perform the unmarshal and marshal operations, and also manipulate the OTD tree structure as a Collaboration might do by using the *Add Instance* and *Delete Instance* buttons. By using these latter features, you can prepare an output data file that can then be used as an input data file for testing purposes.

Clicking the **Tester** icon (see Table 28) in the OTD Editor toolbar saves the OTD to the Repository and displays the OTD Tester as the lower part of the editor, as shown in Figure 80.

Figure 80 OTD Tester



The data display panel on the right has four data display modes, selectable by tabs:

- Input
- Output
- Status
- Verbose

The *Input* panel is displayed by default. Use of the OTD Tester is described in **Using the OTD Tester** on page 132.

Table 29 OTD Tester Buttons

Button	Command	Function
	Run Tester	Runs the tester with the data values entered in the table; the data is unmarshaled into the OTD, and the results are marshaled and displayed in the *Output* panel.
	Add Instance	Adds an an optional node (button is deactivated when an optional node already exists) or an additional instance of the selected repeating node. The repeating node is added immediately following the selected instance. You can also select the length field for a repeating node, in which case the new node will be added as the first instance.
	Delete Instance	Deletes the selected instance of an optional or repeating node. Button is deactivated if the optional instance does not exist.

Next to the OTD Tester buttons is a check box labeled **Show as hex**. When checked, the values for **byte[]** nodes are shown in hexadecimal format (default setting). When not checked, the values are shown as standard alphanumeric text.

Table 30 OTD Tester Icons

Icon	Command	Function
	Open	Displays the *Open File* dialog box, where you select the file you want to open.
	Save	Displays the *Save File* dialog box, where you select the location in which you want to save the file.
	Refresh	Repopulates the OTD object elements with the values from the data display *Input* panel.
	Specify Options	Present only if you are using the extended language options, displays a dialog box in which you specify the testing options (see **Testing Options** on page 129).

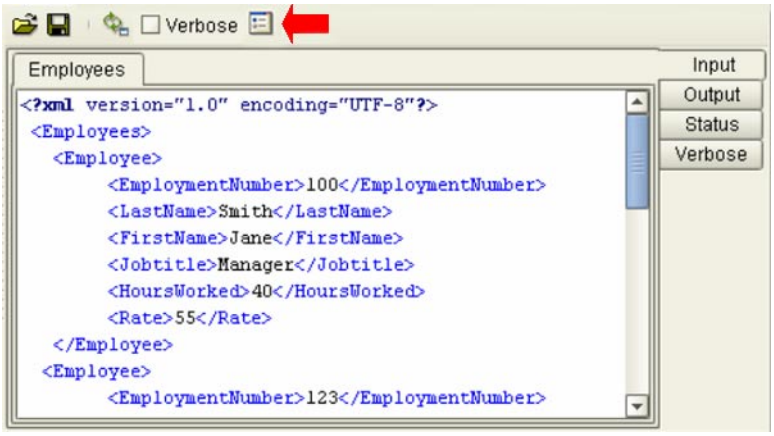
Next to the Collaboration Tester buttons is a check box labeled **Verbose**. Selecting this check box causes parsing information to appear on the *Verbose* panel of the tester.

Testing Options

If you have enabled the extended language options (see **Options Setup** on page 57), an *Options* icon appears in the OTD Tester as shown in Figure 81. This icon allows you to specify:

- The data encoding for the input file of an externally-defined OTD.
- The marshal/unmarshal type for a user-defined OTD.

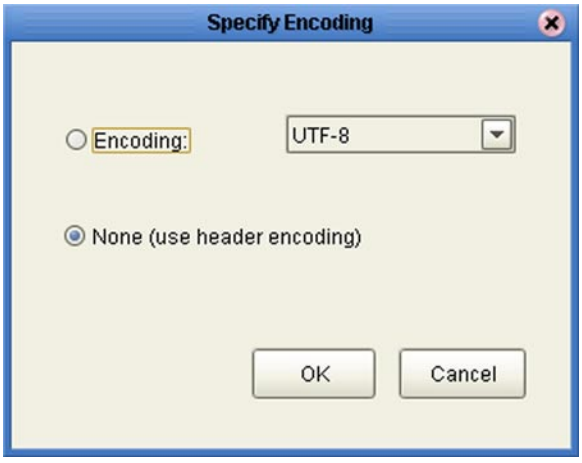
Figure 81 OTD Tester - Data Encoding Enabled



To specify the data encoding for testing an externally-defined OTD

1 Click the *Options* icon above the data display panel (see Figure 81) to display the *Specify Encoding* dialog box shown in Figure 82.

Figure 82 Specify Encoding Dialog Box



- Select **None** (the default setting) to use the header encoding. If the header encoding does not exist, the default locale encoding (based on the Regional and Language Options as set in Windows) is used instead.
- Select **Encoding** to explicitly specify the encoding from a drop-down list, the contents of which depends upon the locale (see Table 31).

Note: *Encoding applies to XML-based data only.*

Table 31 Encoding Options According to Locale

English	Japanese	Korean	Simplified Chinese	Traditional Chinese
UTF-8	UTF-8	UTF-8	UTF-8	UTF-8
ASCII	ASCII	ASCII	GB2312	Big5
	EUC-JP	EUC-KR		
	SJIS			
	MS932			

2 Click **OK** to save your selection.

To specify the marshal/unmarshal type for testing a User-Defined OTD

1 Click the *Options* icon above the data display panel (see Figure 81) to display the *Specify Options* dialog box shown in Figure 83.

Figure 83 Specify Options Dialog Box

2 Select the type of *Unmarshal* and *Marshal* method to use in testing the OTD. The respective defaults are **unmarshal** and **marshal**.

3 Click **OK** to save your selection.

Table 32 Process Differences

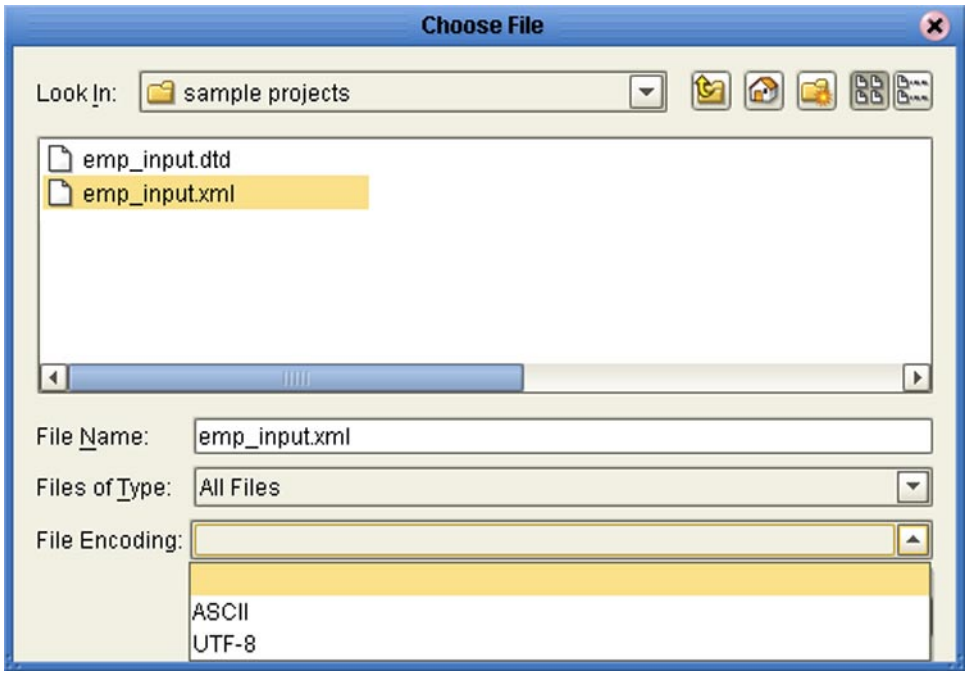
Option	Basic Process
unmarshalFromBytes	antecoding > decoding > parsing
unmarshalFromString	decoding > parsing
marshalToBytes	serialization > encoding > postcoding
marshalToString	serialization > encoding

Note: *See* **Specifying Data Coding** *on page 161 for descriptions of the marshaling and unmarshaling processes and the associated default conditions.*

To specify the data encoding for testing a User-Defined OTD

1 Click either the *Open File* or *Save* icon (whichever is appropriate) above the data display panel to display the file browser dialog box shown in Figure 84. If you are using the extended language options, the *File Encoding* field will be shown and the drop-down list will be enabled.

Figure 84 File Browser - File Encoding Options



2 Display the list and select a File Encoding option to explicitly specify the file encoding. The contents of the list depends upon the locale (see Table 31); the options for English are shown in the figure.

Note: *The encoding specified here is used only to read the file into the tester; marshaling and unmarshaling are accomplished using the encoding specified in the OTD.*

6.3.2 Using the OTD Tester

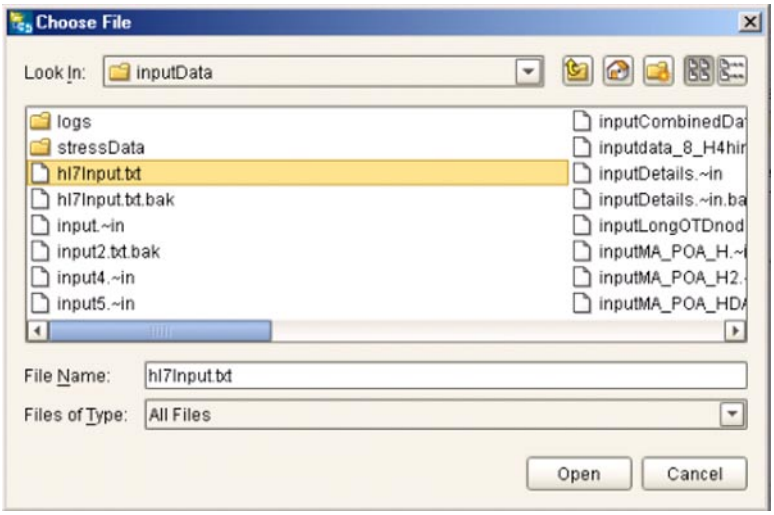
To test data unmarshaling functionality

1 Open or create an OTD.

2 Click the **Tester** icon to display the OTD Tester.

3 Provide the input test data by selecting a data file (see Figure 85).

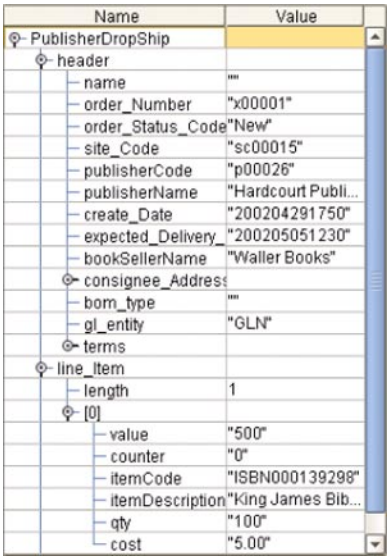
Figure 85 Select Data File



4 Click the **Refresh** button to unmarshal the data.

5 Verify the output by checking the values for each element for correctness (see Figure 86).

Figure 86 Object Elements and Values



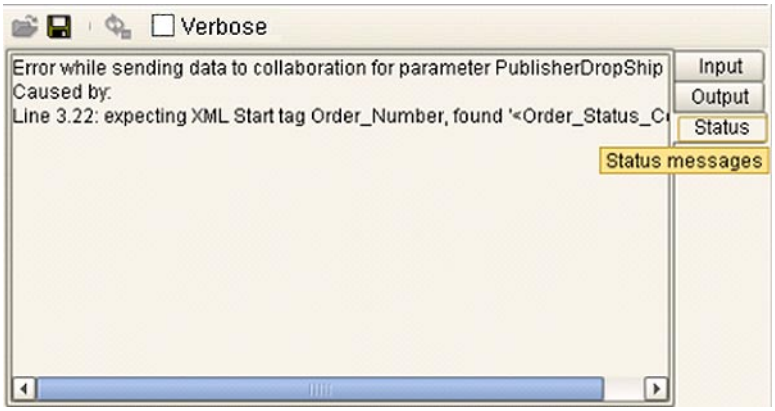
6 Save your input test data to a file for re-use by selecting the *Input* panel and clicking the **Save** icon.

7 You can also change your test data in the *Input* panel, then re-test the OTD by clicking the **Refresh** icon (see **Table 30 on page 128**) to repopulate your OTD object elements with the new values.

Note: *Data, which you can edit for testing, is shown in black type.*

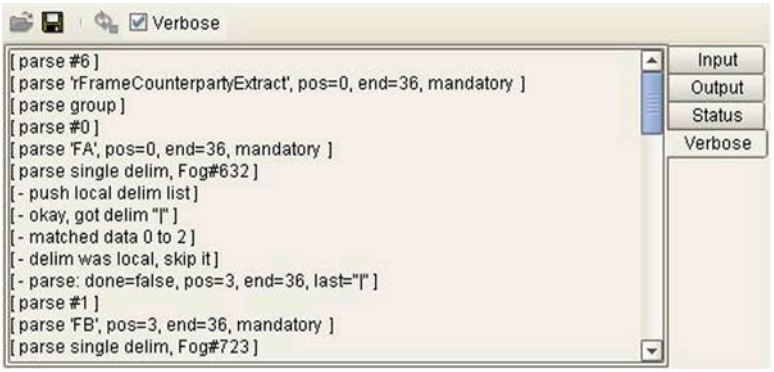
8 If there are errors in your input data, the **Status** panel is automatically invoked, showing the appropriate error messages (see Figure 87).

Figure 87 Data Display - Status Panel



9 For selected OTDs, the **Verbose** option provides a trace of parsing actions during the unmarshal process to aid in debugging the OTD structure. Selecting the **Verbose** check box causes parsing information to appear on the *Verbose* panel (see Figure 88). The format and content of the data display are OTD-specific.

Figure 88 Data Display - Verbose Panel



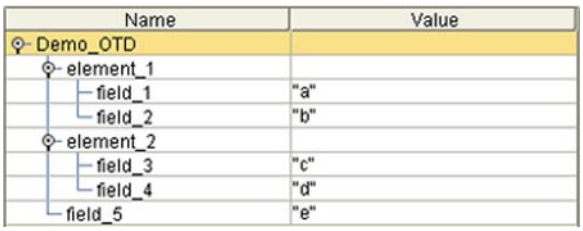
To test data marshaling functionality

1 Open or create an OTD.

2 Click the **Tester** icon to display the OTD Tester.

3 Enter or change data values for each node in the **Value** column of the node table (see Figure 89). Use the Add/Delete Instance (**+**/**-**) buttons to add or remove instances where appropriate.

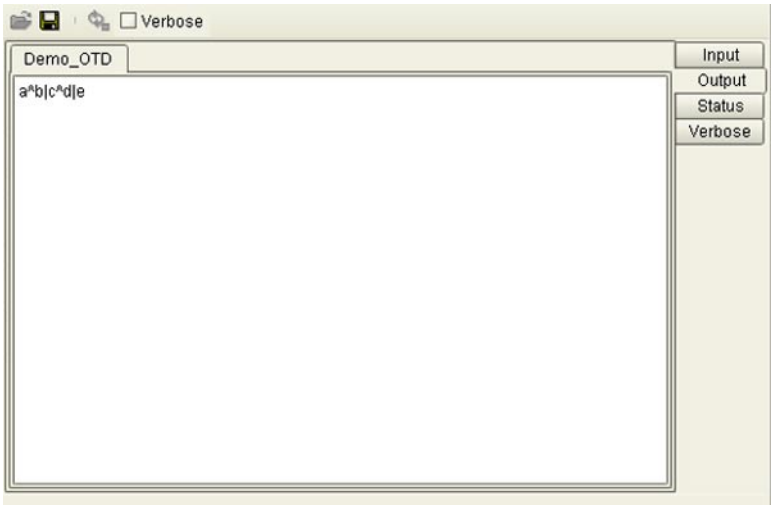
Figure 89 OTD Tester Node Table



4 Click the **Run Tester** button to marshal (serialize) the data.

5 The output is displayed in the *Output* panel (see Figure 90).

Figure 90 Serialized Data in Output Panel



6.4 Externally-Defined OTDs

6.4.1 Using the DTD Wizard

Note: *The **reset()** method resets the DTD OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources.*

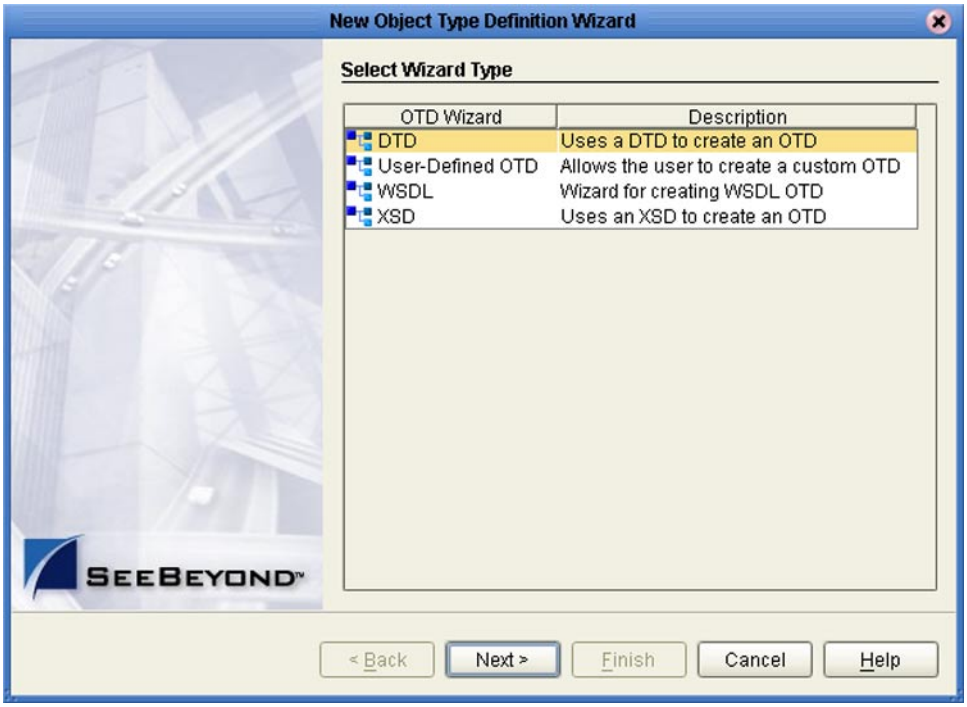
Note: *A newly-created DTD OTD can be redefined by selecting the **Relaunch** option from its context menu in Enterprise Explorer (see* **Object Type Definition Menu** *on page 103**). This relaunches the DTD Wizard, and allows you to reselect files and options while preserving the original OID (imported OTDs cannot be modified in this manner).*

Important: *When using the **Relaunch** option, the node structure of the OTD must not change.*

To create an OTD file from a DTD file

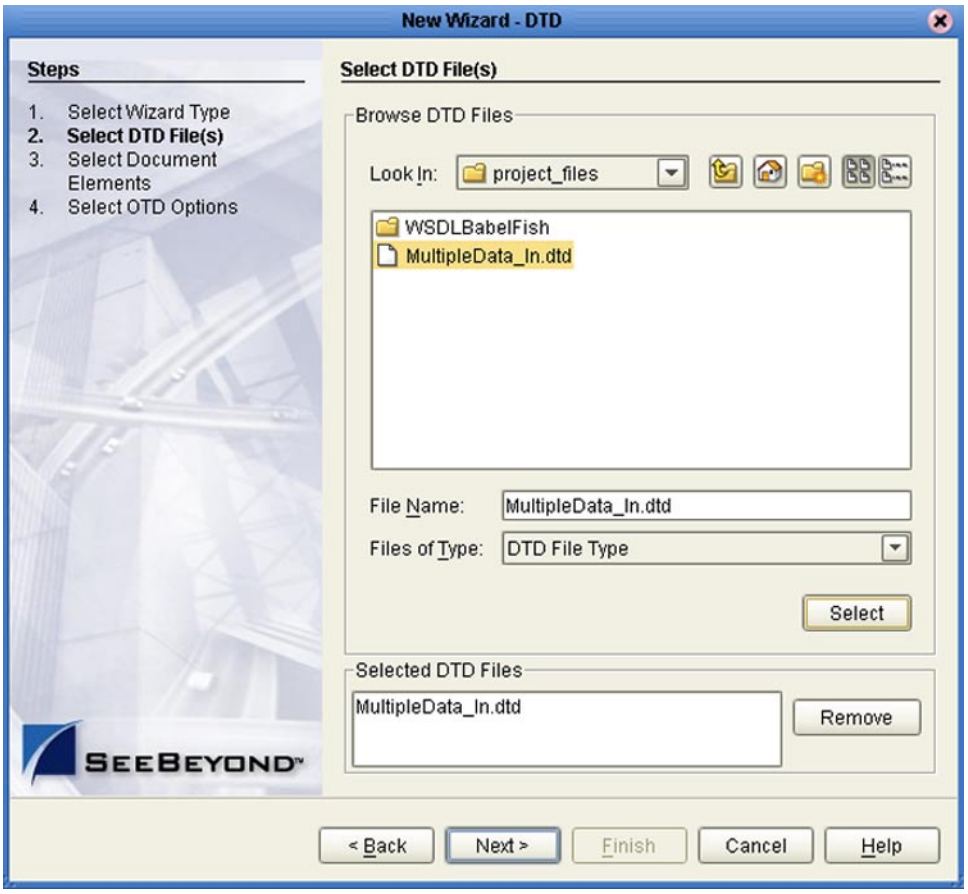
1 In the *Select Wizard Type* dialog, select **DTD** from the *OTD Wizard* list (see Figure 91) to create an OTD from a Data Type Definition (DTD) file.

Figure 91 OTD Wizard Selection: DTD Wizard



2 Click **Next** to display the *Select DTD File(s)* dialog box, shown in Figure 92.

Figure 92 Select DTD File(s) Dialog Box



3 In the *Look In* drop-down list, navigate to the DTD file or files that you want to use to create the OTD. Click **Select** to add the files to the *List of Selected DTDs*.

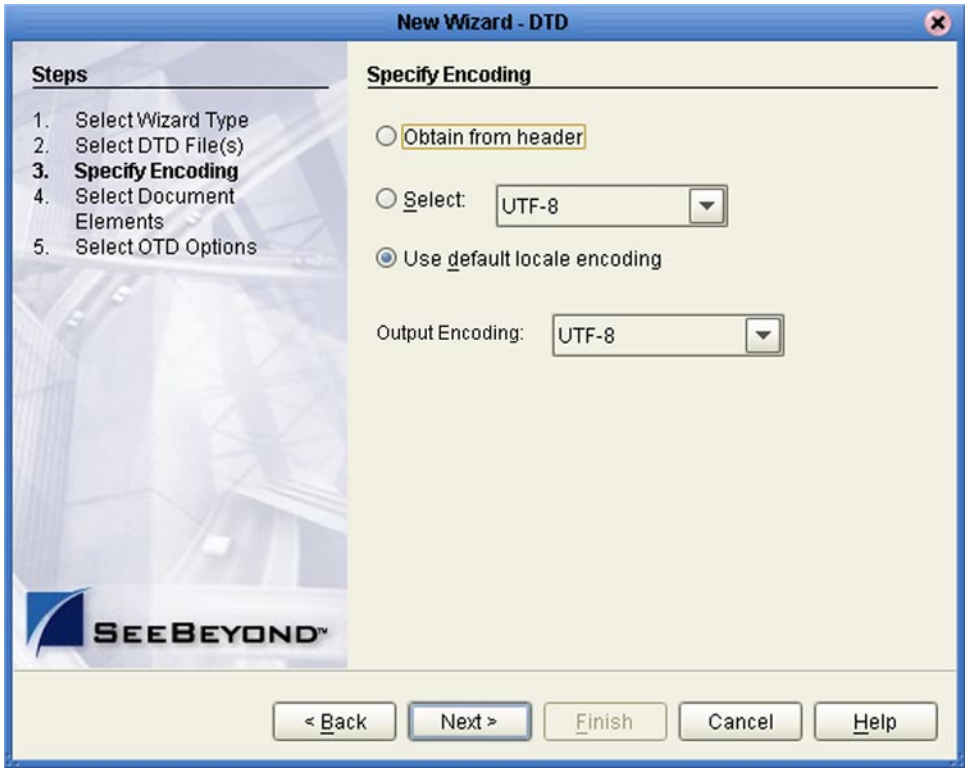
- If the DTD file does not contain all information required for building an OTD (such as element definitions) a warning box such as that shown in Figure 93 will be displayed.

Figure 93 Cannot Create OTD Warning Box

4 If you are using the extended language options (see **Options Setup** on page 57) click **Next** to display the *Specify Encoding* dialog box, shown in Figure 94. If you are *not* using the extended language options, this Wizard dialog will not appear—proceed to step 7.

Note: *This OTD encoding is currently supported for Java Collaboration Definitions only.*

Figure 94 Select Encoding Dialog Box



5 Click the appropriate option button to specify the data encoding for the input XML file:

- **Obtain from header** does not apply to DTD OTDs.
- Click **Select** to explicitly specify the encoding from a drop-down list, the contents of which depends upon the locale (see Table 33).

Table 33 Encoding Options According to Locale

English	Japanese	Korean	Simplified Chinese	Traditional Chinese
UTF-8	UTF-8	UTF-8	UTF-8	UTF-8
ASCII	ASCII	ASCII	GB2312	Big5
	EUC-JP	EUC-KR		
	SJIS			
	MS932			

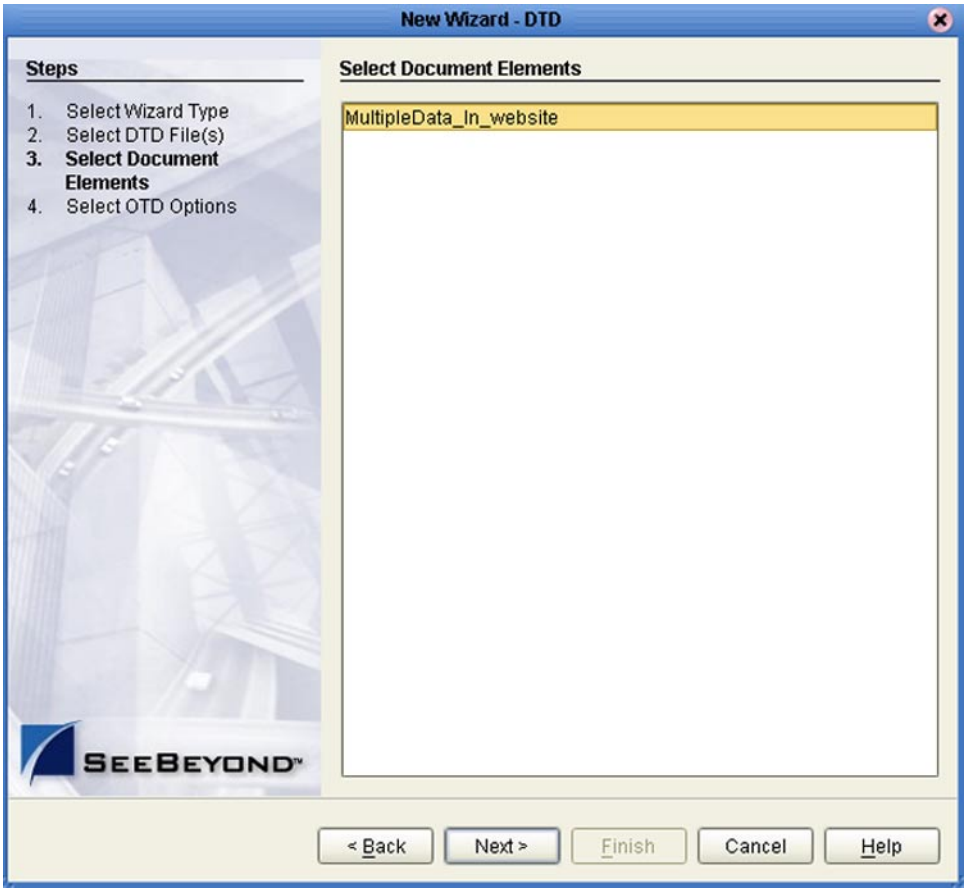
- Select **Use default locale encoding** to use the default encoding for the locale.

6 Select **Output Encoding** to specify the header encoding of the output XML file from a drop-down list, the contents of which depends upon the locale (see Table 33).

Note: *For the Output Encoding property to apply to an imported project from a previous eGate Integrator release, you must recreate the OTD and specify the new encoding option.*

7 Click **Next** to display the *Select Document Elements* dialog box, shown in Figure 95.

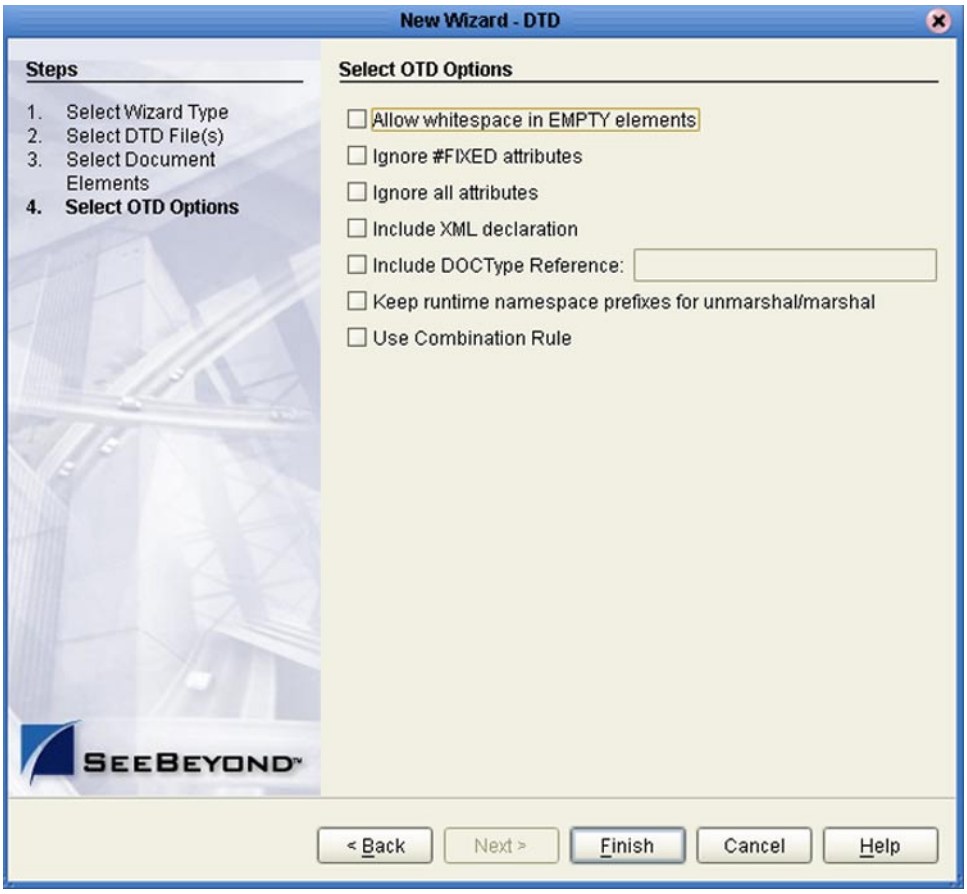
Figure 95 Select Document Elements Dialog Box



8 Select the elements of the document that you want to include in the OTD.

9 Click **Next** to display the *Select OTD Options* dialog box, shown in Figure 96.

Figure 96 Select OTD Options Dialog Box



10 Select the check boxes next to the OTD options you want to enable (see Table 34).

Table 34 DTD OTD Options

Option	Description
Allow whitespace in EMPTY elements	If an element is defined as EMPTY, this option controls whether or not white spaces are allowed within the element in the XML instance document.
Ignore #FIXED attributes	This option controls whether or not attributes defined as FIXED are ignored during the unmarshal and marshal processes. ▪ If this option is *not* selected, the attribute is recognized and saved into the OTD's runtime structure during the unmarshal process, and also appears in the output during the marshal process. ▪ If this option *is* selected, the attribute is ignored and neither of the above occurs.

Option	Description
Ignore all attributes	This option controls whether or not all attributes are ignored during the unmarshal and marshal processes. If both this option and the *Keep runtime namespace prefixes…* option (below) are selected, only namespace attributes will be handled during the unmarshal process and consequently presented in the output during the marshal process. (The *namespace* attribute has the form **xmlns:XX**.)
Include XML declaration	This option controls whether or not the XML declaration **<?xml version="1.0" encoding="……"?>** appears in the output during the marshal process.
Include DOC Type Reference	This option controls whether or not the **"<!DOCTYPE …"** string appears in the output during the marshal process.
Keep runtime namespace prefixes for unmarshal/ marshal	This option controls whether or not the namespace prefixes used during the marshal process are identical to those used in the unmarshal process. ▪ If this option is selected, all namespace attributes will be preserved once they appear in the XML instance document, and the namespace prefixes used in the marshal process will be exactly as they were presented in the XML document during the unmarshal process. ▪ If this option is *not* selected, then the namespace prefixes used in the marshal process might be different than the ones presented in the XML document during the unmarshal process (for example, the namespace prefixes that are presented in the XSD file might be used). **Note:** A consequence of selecting this option is that if there is no unmarshal process performed before the marshal process, then there will be no namespace attributes presented in the output (see the comment for the option below).
Use Combination Rule	Not currently used.

11 Click **Finish** to add the OTD to your Project and invoke the OTD Editor.

6.4.2 Using the WSDL Wizard

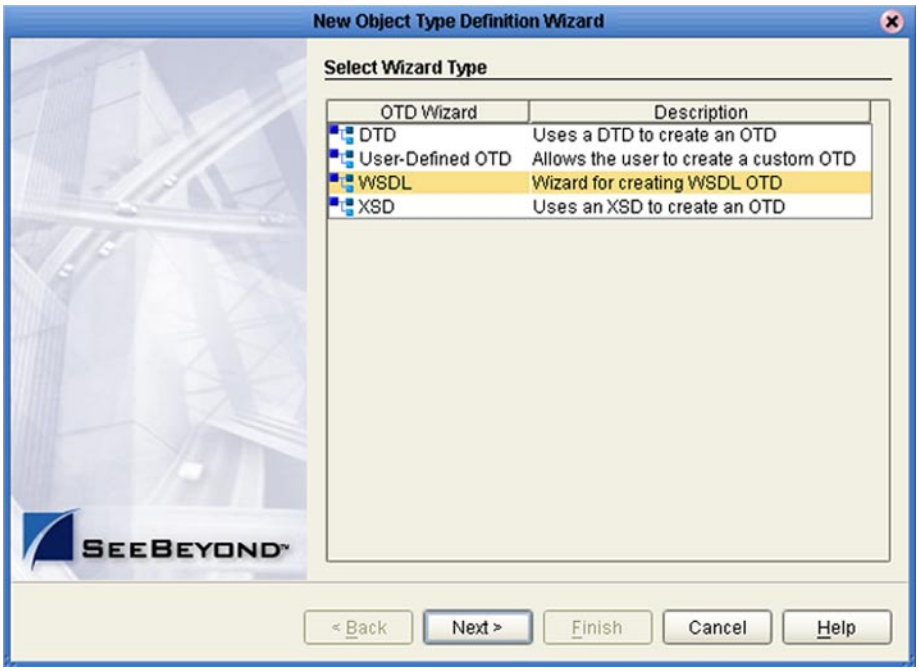
Note: *A newly-created WSDL OTD can be redefined by selecting the* ***Relaunch*** *option from its context menu in Enterprise Explorer (see* **Object Type Definition Menu** *on page 103). This relaunches the WSDL Wizard, and allows you to reselect files and options while preserving the original OID (imported OTDs cannot be modified in this manner).*

Important: *When using the* ***Relaunch*** *option, the node structure of the OTD must not change.*

To create an OTD file from a WSDL file

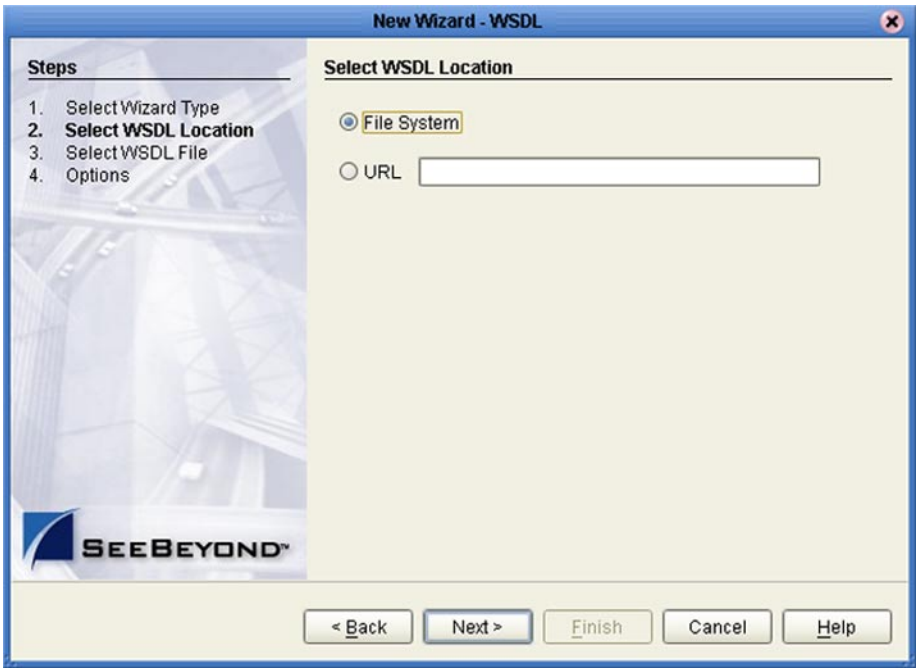
1 In the *Select Wizard Type* dialog, select **WSDL** from the *OTD Wizard* list (see Figure 97) to create an OTD from an WSDL file.

Figure 97 OTD Wizard Selection: WSDL Wizard



2 Click **Next** to display the *Select WSDL File Location* dialog, shown in Figure 98

Figure 98 WSDL Wizard: Select WSDL Location

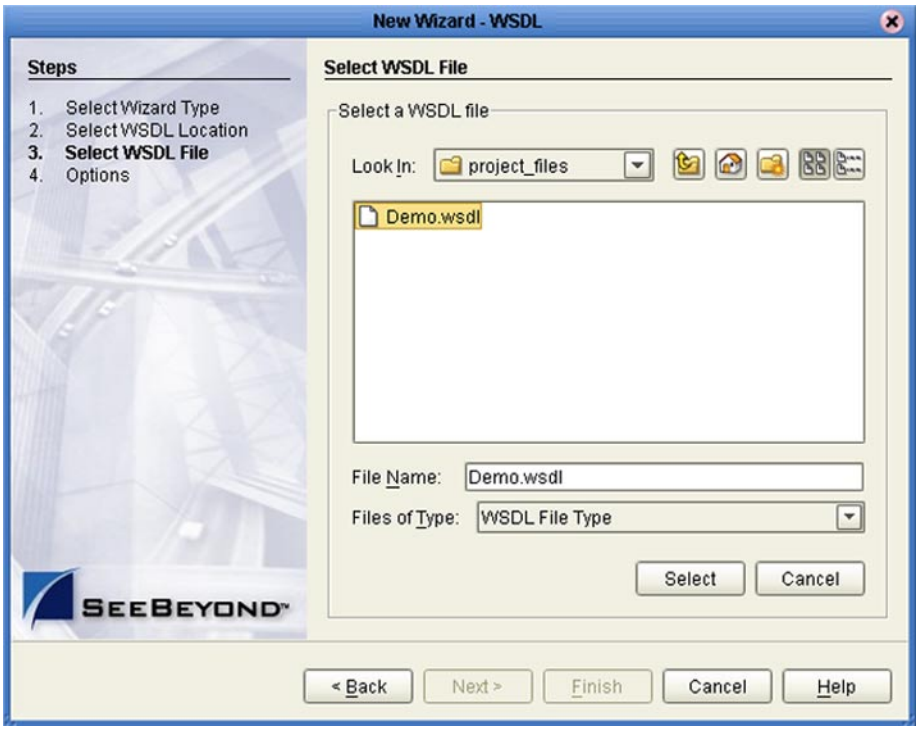


3 In the *Select WSDL Location* dialog, select **File System** or enter a **URL**, depending upon where your WSDL file is located.

Note: *If you are using a URL that points outside your firewall, you must set up a proxy server (see* **Using a Proxy Server** *on page 51).*

4 Click **Next** to display the *Select WSDL File* dialog, shown in Figure 99.

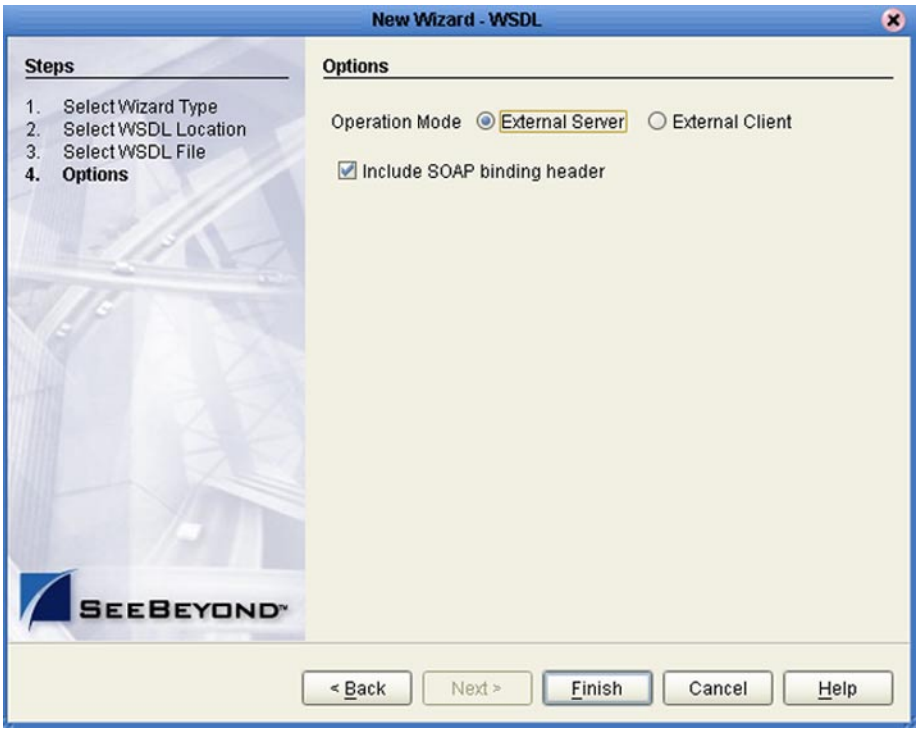
Figure 99 WSDL Wizard: Select WSDL File



5 In the *Look In* drop-down list, navigate to the WSDL file or files that you want to use to create the OTD. Click **Select** to add the files to the *List of Selected WSDLs*.

6 Click **Next** to display the *Options* dialog, shown in Figure 100.

Figure 100 WSDL Wizard: Select OTD Options



7 Select the required Operational Mode:

- If you are building an ICAN Client for an external Web Service, select **External Server**.
- If you are implementing and exposing an ICAN Web Service, select **External Client**.

8 If you want to include SOAP header elements in the SOAP envelope at runtime (see **SOAP Header Elements** on page 146), select the check box for **Include SOAP binding header**.

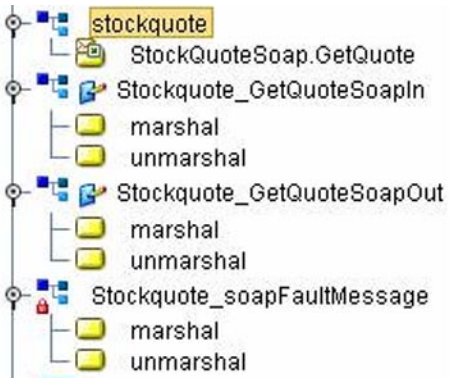
Note: *To enable this option, the* ***wsdl:binding*** *section in the WSDL document selected in Wizard Step 3 must contain the necessary binding information for* ***soap:header****.*

9 Click **Finish** to add the OTD to your Project and invoke the OTD Editor.

WSDL OTD Structure

The WSDL OTD has the following basic structure, as viewed in the Project Explorer:

Figure 101 Typical WSDL OTD Structure



WSDL Operation Elements

To tie your messages together as a request-response pair corresponding to a method call, you must define operations using the WSDL **<operation>** element. A WSDL operation identifies messages as being *input* or *output*.

Inside the WSDL file's **<operation>** element**,** you specify your **<input>** and **<output>** elements. Each element refers to the corresponding message by its fully-qualified name. The collection of all WSDL operations (that is, methods) exposed by your service is called a **portType** and is defined using the WSDL **<portType>** element. The OTD structure may be depicted as:

Root Node
PortType_XXX
Operation_XXX
Input_XXX
Output_XXX

Where **XXX** is the name for each element given in the original WSDL file.

The **<operation>** element is a child of **<portType>**. You can name the **<portType>** whatever you want. The port type **name** attribute provides a unique name among all the PortTypes defined within the enclosing WSDL file. Each WSDL operation is named via the **name** attribute.

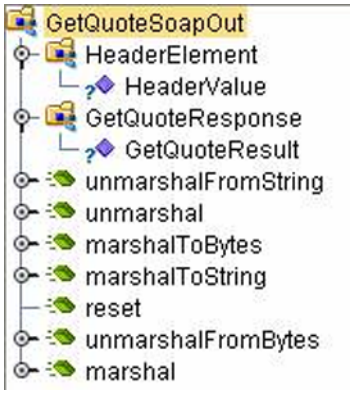
Each operation within a WSDL OTD (like its WSDL file counterpart) uses one of the following operation modes for communication:

- **One-way:** The server receives a message from the client; also referred to as *fire and forget*.
- **Request-response:** The server receives a message from the client and sends a responding message back.

SOAP Header Elements

When a WSDL OTD is created with the *Include SOAP header* option enabled, the input and output messages include a SOAP header element as shown in Figure 102. If the option is not checked, the header element is missing.

Figure 102 Output Message with SOAP Header Included



6.4.3 Using the XSD Wizard

Note: *The **reset()** method resets the XSD OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources.*

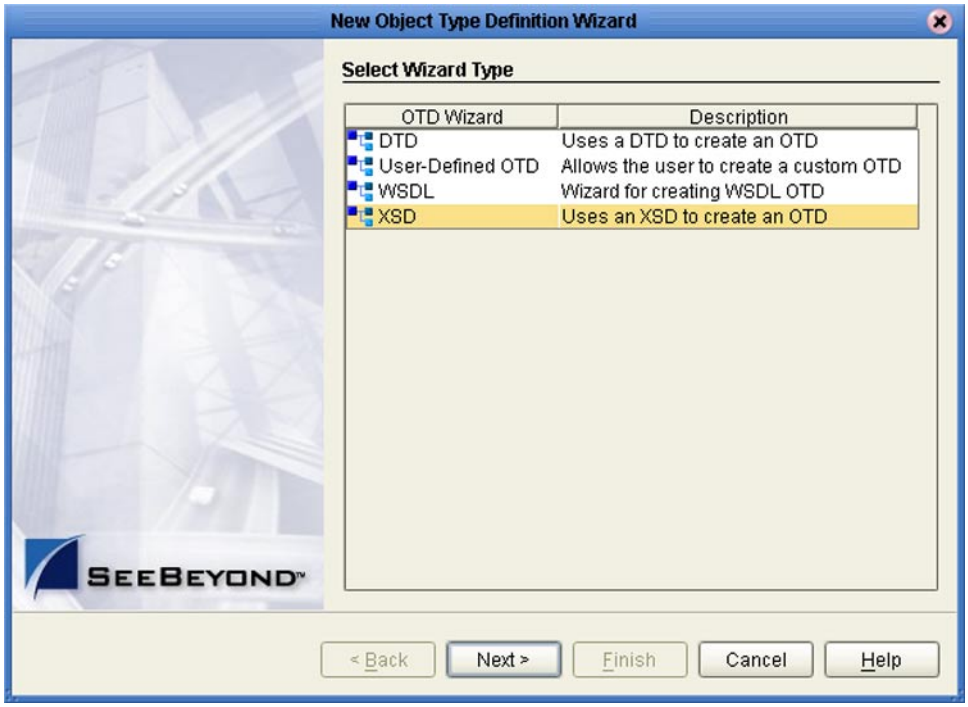
Note: *A newly-created XSD OTD can be redefined by selecting the **Relaunch** option from its context menu in Enterprise Explorer (see* **Object Type Definition Menu** *on page 103**). This relaunches the XSD Wizard, and allows you to reselect files and options while preserving the original OID (imported OTDs cannot be modified in this manner).*

Important: *When using the **Relaunch** option, the node structure of the OTD must not change.*

To create an OTD file from an XSD file

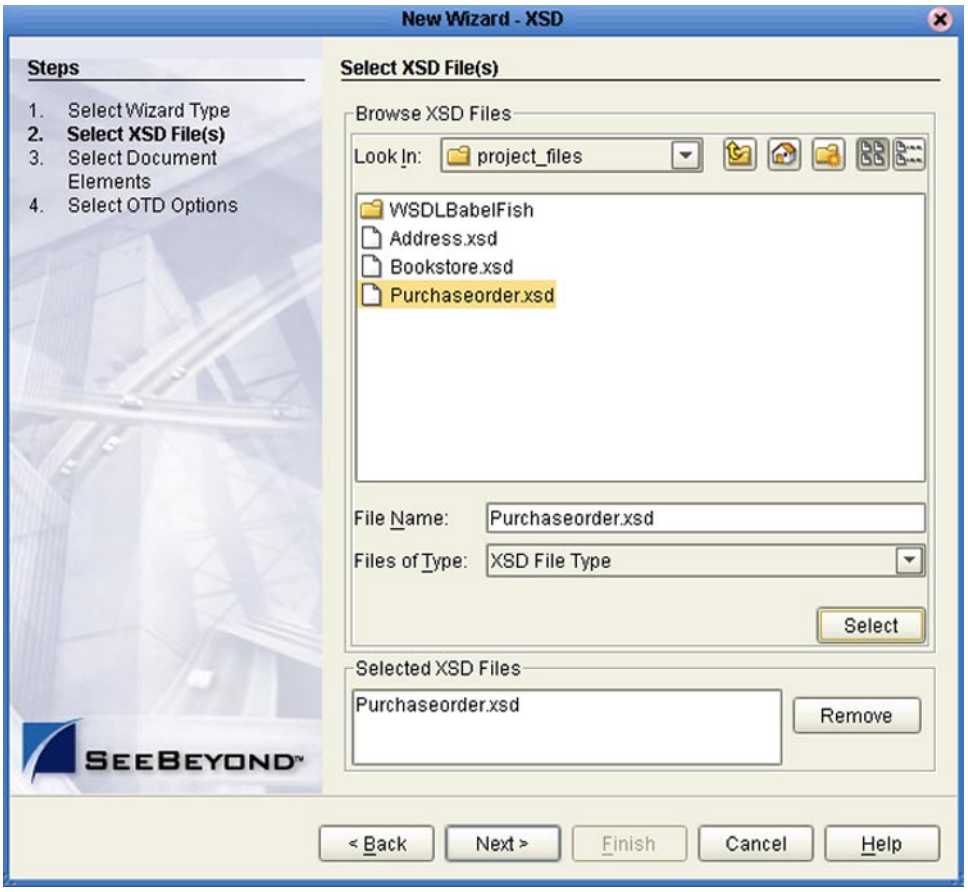
1 In the *Select Wizard Type* dialog, select **XSD** from the *OTD Wizard* list (see Figure 103) to create an OTD from an XSD file.

Figure 103 OTD Wizard Selection: XSD Wizard



2 Click **Next** to display the Select XSD File(s) dialog box, shown in Figure 104.

Figure 104 XSD Wizard: Select XSD File(s)



3 In the *Look In* drop-down list, navigate to the XSD file or files that you want to use to create the OTD. Click **Select** to add the files to the *List of Selected XSDs*.

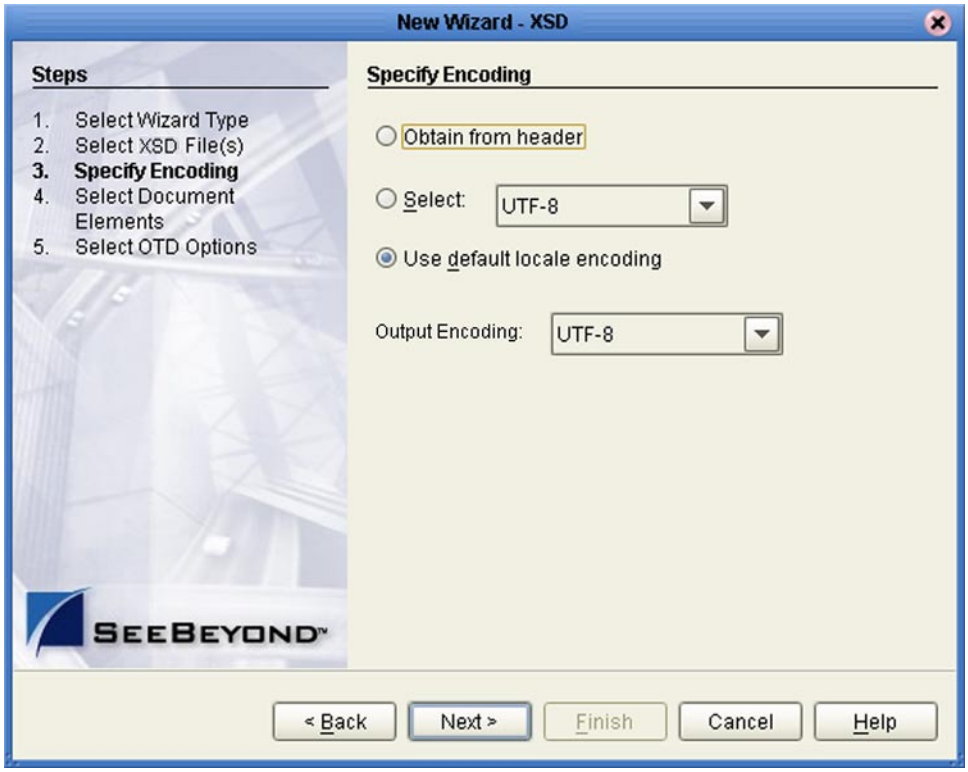
- If the XSD file does not contain all information required for building an OTD (such as element definitions) a warning box such as that shown in Figure 105 will be displayed.

Figure 105 Cannot Create OTD Warning Box

4 If you are using the extended language options (see **Options Setup** on page 57) click **Next** to display the *Specify Encoding* dialog box, shown in Figure 106. If you are *not* using the extended language options, this Wizard dialog will not appear—proceed to step 7.

Note: *This OTD encoding is currently supported for Java Collaboration Definitions only.*

Figure 106 Select Encoding Dialog Box



5 Click the appropriate option button to specify the data encoding for the input XML file:

- Select **Obtain form header** (the default setting) to use the XML header encoding. If the header encoding does not exist, the default locale encoding is used instead.
- Click **Select** to explicitly specify the encoding from a drop-down list, the contents of which depends upon the locale (see Table 35).

Table 35 Encoding Options According to Locale

English	Japanese	Korean	Simplified Chinese	Traditional Chinese
UTF-8	UTF-8	UTF-8	UTF-8	UTF-8
ASCII	ASCII	ASCII	GB2312	Big5
	EUC-JP	EUC-KR		
	SJIS			
	MS932			

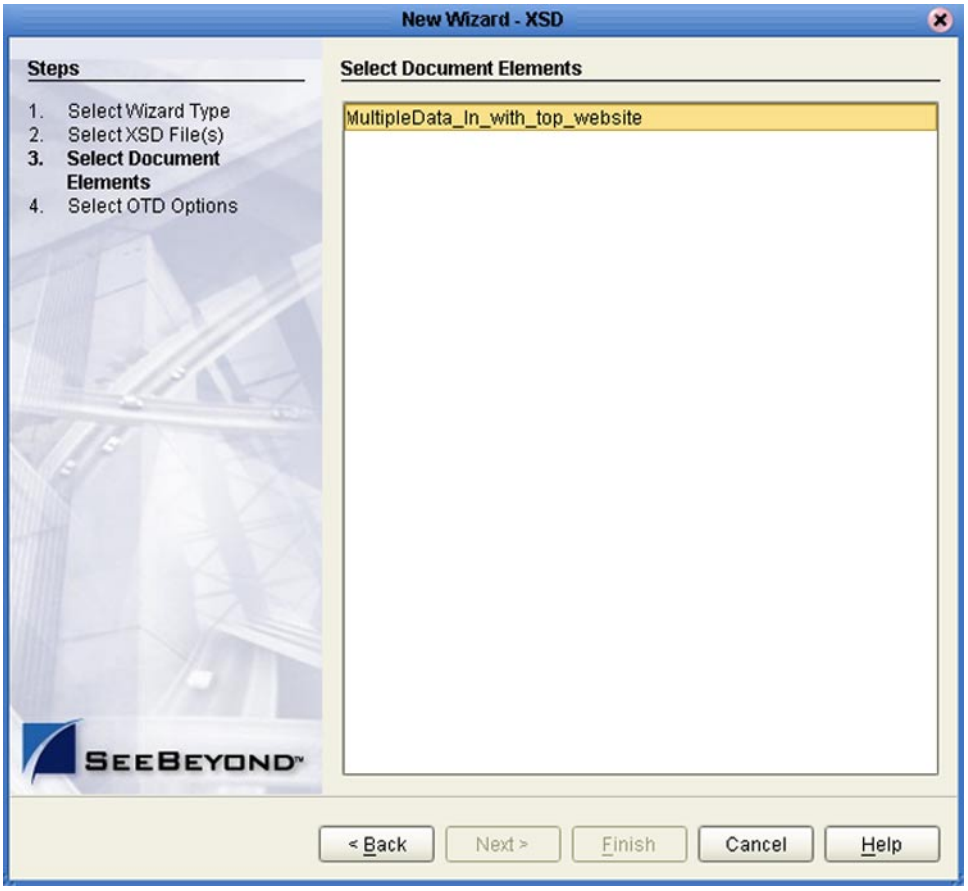
- Select **Use default locale encoding** to use the default encoding for the locale.

6 Select **Output Encoding** to specify the header encoding of the output XML file from a drop-down list, the contents of which depends upon the locale (see Table 35).

Note: *For the Output Encoding property to apply to an imported project from a previous eGate Integrator release, you must recreate the OTD and specify the new encoding option.*

7 Click **Next** to display the *Select Document Elements* dialog box, shown in Figure 107.

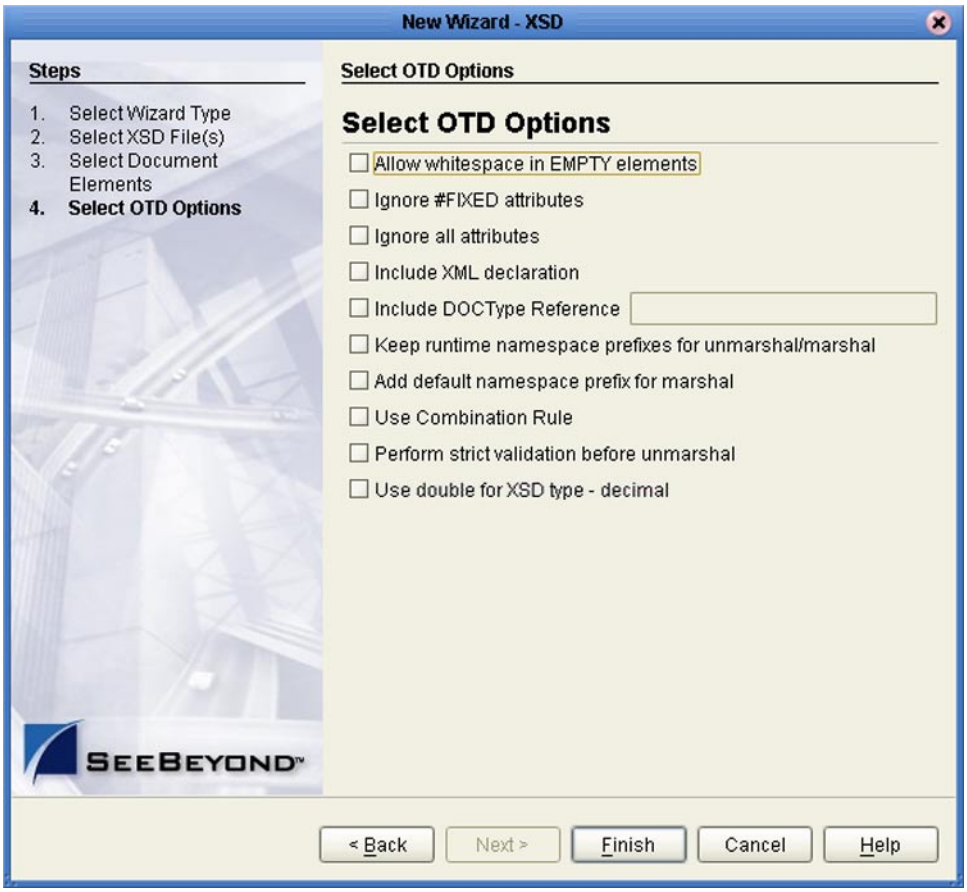
Figure 107 Select Document Elements Dialog Box



8 Select the elements of the document that you want to include in the OTD.

9 Click **Next** to display the *Select OTD Options* dialog box, shown in Figure 108.

Figure 108 Select OTD Options Dialog Box



10 Select the check boxes next to the OTD options you want to enable (see Table 36).

Table 36 XSD OTD Options

Option	Description
Allow whitespace in EMPTY elements	Not currently used for XSD OTDs.
Ignore #FIXED attributes	This option controls whether or not attributes defined as FIXED are ignored during the unmarshal and marshal processes. ▪ If this option is *not* selected, the attribute is recognized and saved into the OTD's runtime structure during the unmarshal process, and also appears in the output during the marshal process. ▪ If this option *is* selected, the attribute is ignored and neither of the above occurs.

Option	Description
Ignore all attributes	This option controls whether or not all attributes should be ignored in the unmarshal and marshal processes. If both this option and the *Keep runtime namespace prefixes…* option (below) are selected, only namespace attributes will be handled during the unmarshal process and consequently presented in the output during the marshal process. (The *namespace* attribute has the form **xmlns:XX**.)
Include XML declaration	This option controls whether or not the XML declaration **<?xml version="1.0" encoding="……"?>** appears in the output during the marshal process.
Include DOC Type Reference	Not currently used for XSD OTDs.
Keep runtime namespace prefixes for unmarshal/ marshal	This option controls whether or not the namespace prefixes used during the marshal process are identical to those used in the unmarshal process. ▪ If this option is selected, all namespace attributes will be preserved once they appear in the XML instance document, and the namespace prefixes used in the marshal process will be exactly as they were presented in the XML document during the unmarshal process. ▪ If this option is *not* selected, then the namespace prefixes used in the marshal process might be different than the ones presented in the XML document during the unmarshal process (for example, the namespace prefixes that are presented in the XSD file might be used). **Note:** A consequence of selecting this option is that if there is no unmarshal process performed before the marshal process, then there will be no namespace attributes presented in the output (see the comment for the option below).
Add default namespace prefix for marshal	This option controls whether or not the prefix of the default target namespace of an element is applied to the element during the marshal process. ▪ If both this flag and the *Keep runtime namespace prefixes…* option (above) are selected, then the default target namespace of an element will be applied to the element during the marshal process, *if it is a root element*. ▪ If the *Keep runtime namespace prefixes…* option is *not* selected, then the elements are qualified based on the XSD definition and this flag has no effect.
Use Combination Rule	Not currently used.
Perform strict validation before unmarshal	Not currently used.
Use double for XSD type - decimal	If this option is checked, Enterprise Designer will use Java double type for the decimal type in the OTD. If not checked (default) Java BigDecimal type is used, which can handle big decimals with high precision.

11 Click **Finish** to add the OTD to your Project and invoke the OTD Editor.

6.5 User-Defined OTDs

When an externally-defined OTD is not a viable option, you can define your own OTD as described in this section.

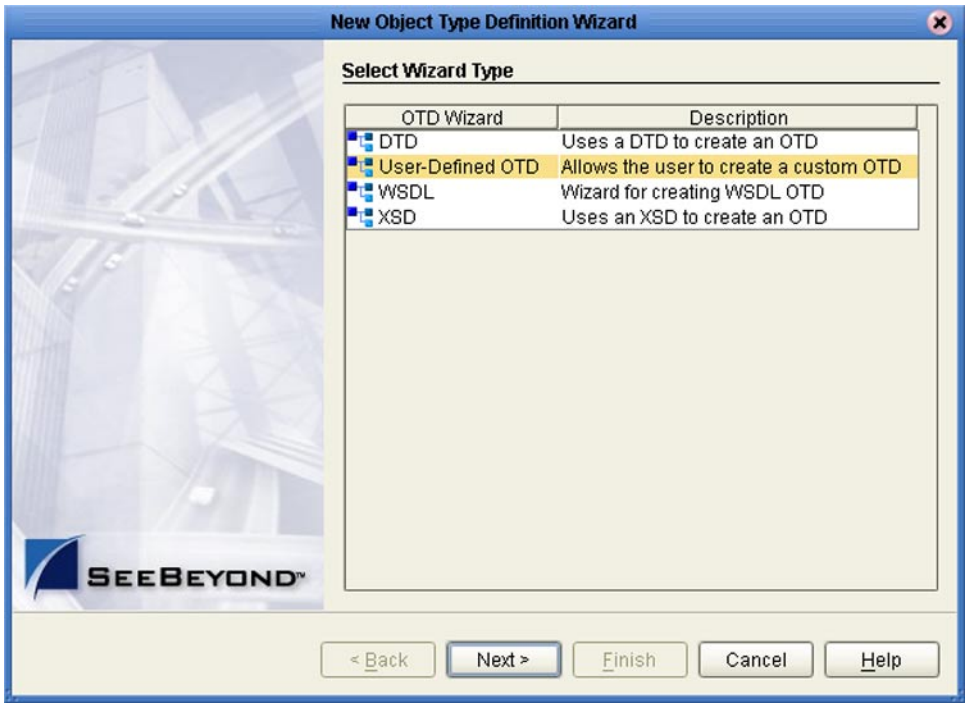
Note: *The **reset()** method resets the User-Defined OTD to its initial conditions, including any default values. It is highly recommended that this method be used in your Collaborations whenever marshaling in a loop, to conserve system resources. (Logical Hosts running on HP NonStop systems will show a dramatic improvement in performance when the **reset()** method is properly used.)*

6.5.1 Using the User-Defined OTD Wizard

To create a User-Defined OTD

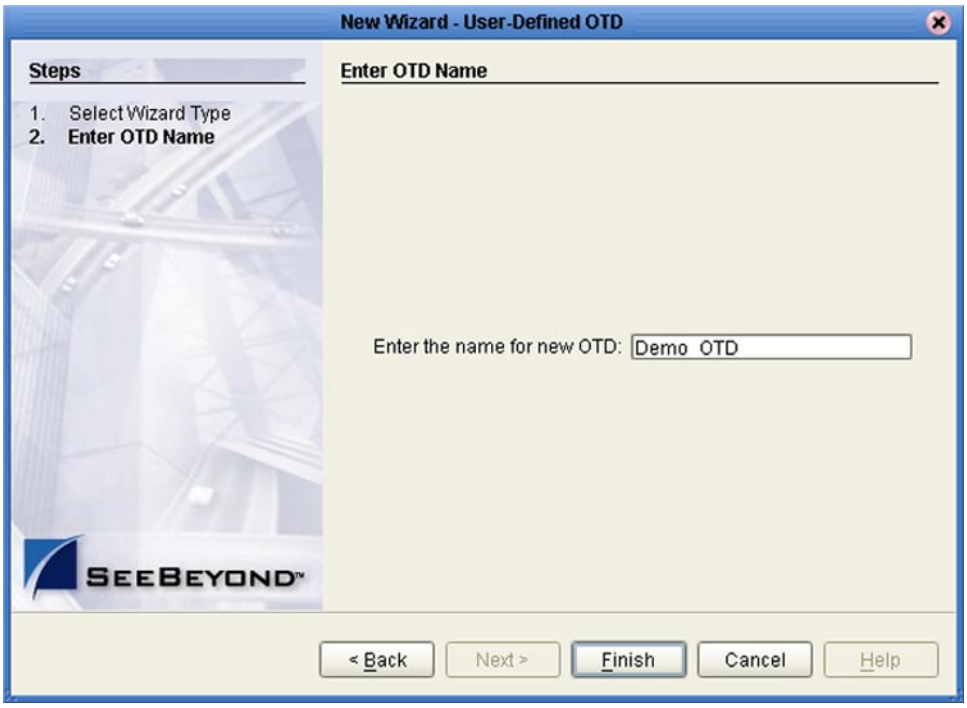
1 In the *Select Wizard Type* dialog, select **User-Defined OTD** from the *OTD Wizard* list (see Figure 109) to create an OTD without using a source file.

Figure 109 OTD Wizard Selection: User-Defined OTD



2 Click **Next** to display the *Enter OTD Name* dialog box, shown in Figure 110.

Figure 110 Enter OTD Name



3 Enter a name for the OTD into the text box provided, then click **Finish** to add the OTD to your Project.

4 The OTD Editor will open, displaying the new OTD (see **OTD Editor** on page 125). You now must specify the OTD in detail, as described in the following sections.

6.5.2 Creating and Managing Nodes

The OTD Editor allows you to:

- **Rename** a User-Defined OTD node.

  Renaming the root node renames the OTD.

- **Add** element and field nodes to a User-Defined OTD.
- **Delete** root, element, and field nodes from a User-Defined OTD.

  When a node is *deleted*, both the node and its associated 'children' are deleted.

- **Prune** root nodes in a User-Defined OTD.

  When a node is *pruned*, only its associated 'children' are deleted, while the node itself is preserved. Pruning can only be performed on root nodes.

- **Configure** the User-Defined OTD node properties.

Note: *If you move an OTD node, you must reset the* ***nodeType*** *for that node. See* **Editing OTD Properties** *on page 157.*

Creating and Modifying Elements

Once they are created, elements can be renamed or deleted, and their properties can be configured as described in **Element Node Properties** on page 159. Elements are created and placed as described below.

To add an Element to an OTD

1 Select the node of interest.

2 Right-click to invoke the node's context menu.

3 Click **Add Element**, which displays the options shown in Figure 111 and described in Table 37.

Figure 111 Add Element Options

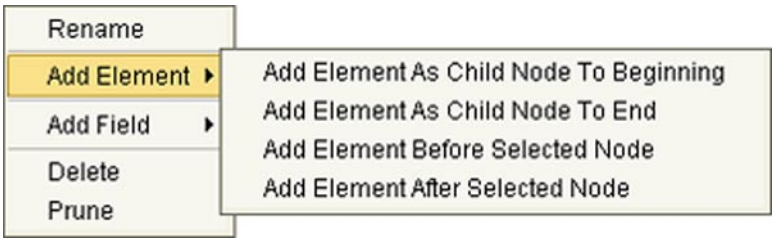
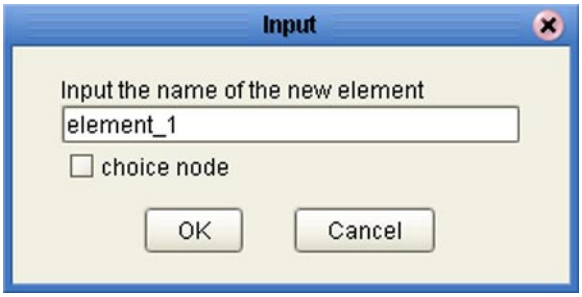


Table 37 Add Element Options

Option	Description
Add Element as a Child Node to Beginning	Adds an Element as a child of the selected node, before any other children.
Add Element as a Child Node to End	Adds an Element as a child of the selected node, after any other children.
Add Element Before Selected Node	Adds an Element at the same level as the selected node, above the selected node.
Add Element After Selected Node	Adds an Element at the same level as the selected node, below the selected node.

4 Select the desired option, which invokes the **Input** dialog box shown in Figure 112.

Figure 112 Input Dialog Box



5 Type in the name for the element.

6 If you want the element to be a *choice* node, check the box provided.

7 Click **OK** to complete the addition of the element to the OTD.

Note: *A **choice** node is either an alternate or transient Node Type. See* **Specifying the Node Type** *on page 167 for more information.*

Creating and Modifying Fields

Once they are created, fields can be renamed or deleted, and their properties can be configured as described in **Field Node Properties** on page 160. Fields are created and placed as described below.

To add a Field to an OTD

1 Select the node of interest.

2 Right-click to invoke the node's context menu.

3 Click **Add Field**, which displays the options shown in Figure 113 and described in Table 38.

Figure 113 Add Field Options

Table 38 Add Field Options

Option	Description
Add Field as a Child Node to Beginning	Adds a Field as a child of the selected node, before any other children.
Add Field as a Child Node to End	Adds a Field as a child of the selected node, after any other children.
Add Field Before Selected Node	Adds a Field at the same level as the selected node, above the selected node.
Add Field After Selected Node	Adds a Field at the same level as the selected node, below the selected node.

4 Select the desired option, which invokes an **Input** dialog box similar to that shown in Figure 112.

6.5.3 Editing OTD Properties

Note: *Clicking a **Value** field selects it, after which you must double-click to open it for editing.*

Root Node Properties

The set of properties associated with the root node is shown in Figure 114.

Figure 114 User-Defined OTD Root Node Properties

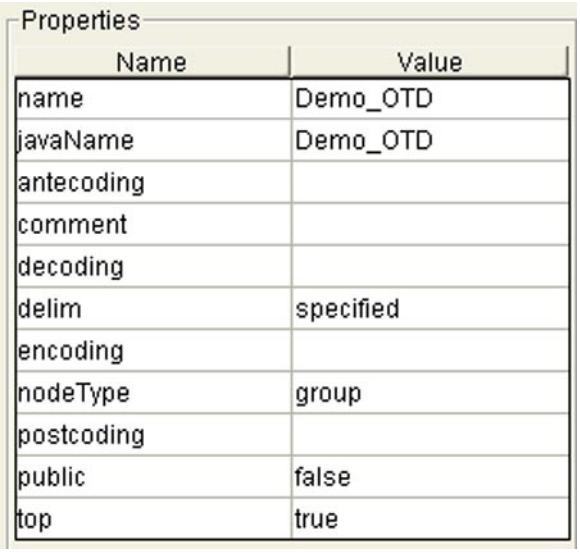


Table 39 Root Node Properties

Name	Description
name	Node display name (see **Overview** on page 119).
javaName	Property accessor basename, automatically generated—*do not modify*. See **Overview** on page 119.
antecoding	Specifies the *input data* coding (see Table 40 and **Specifying Data Coding** on page 161). If this property is not specified, the value specified for the **decoding** property will be used for the input data.
comment	Free-form text (no run-time effect).
decoding	Specifies the *unmarshal* coding (see Table 40 and **Specifying Data Coding** on page 161). (It is recommended to use UTF-8 for DBCS data, since the hex value of some ASCII delimiter may coincide with a hex value contained within a double-byte character.)
delim	Delimiter specification (see **Specifying Delimiters** on page 168). Default value is **not set**.
encoding	Specifies the *marshal* coding (see Table 40 and **Specifying Data Coding** on page 161).

Name	Description
nodeType	Specifies the marshal/unmarshal format (see **Specifying the Node Type** on page 167). The default value is **group**.
postcoding	Specifies the *output data* coding (see Table 40 and **Specifying Data Coding** on page 161). If this property is not specified, the value specified for the **encoding** property will be used for the output data.
public	For future use, not currently active; **false** by default.
showDelim	Displayed only if **nodeType** is *delim*. Once delimiters are specified, value field shows the delimiters. (Root nodes are not typically delimited.)
top	Specifies whether or not marshal/unmarshal is supported (**true** or **false**).

Table 40 Encoding Options According to Locale

English	Japanese	Korean	Simplified Chinese	Traditional Chinese
UTF-8	UTF-8	UTF-8	UTF-8	UTF-8
ASCII	ASCII	ASCII	GB2312	Big5
	EUC-JP	EUC-KR		
	SJIS	MS949		
	MS932			

Element Node Properties

The set of properties associated with element nodes is shown in Figure 115.

Figure 115 User-Defined OTD Element Node Properties

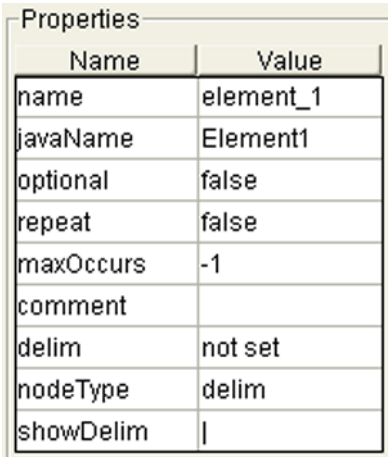


Table 41 Element Node Properties

Name	Description
name	Element display name.
javaName	Property accessor basename, automatically generated—*do not modify*.
optional	Specifies whether or not the node can be absent (**true** or **false**). Not applicable if the element is the child of a *choice* node.
repeat	Specifies whether or not the node can appear multiple times (**true** or **false**). Not applicable if the element is the child of a *choice* node.
maxOccurs	Specifies the maximum number of occurrences of the node if the node is repeating. Property has no effect if node is non-repeating, but may show error during validation if set to value >1.
comment	Free-form text (no run-time effect).
delim	Delimiter specification (see **Specifying Delimiters** on page 168).
nodeType	Governs the marshal/unmarshal format. The default value is **delim**, unless the element is a *choice* node—then the default is **trans**.
showDelim	Displayed only if **nodeType** is *delim*. Once delimiters are specified, value field shows the delimiters.

Field Node Properties

The set of properties associated with field nodes is shown in Figure 116.

Figure 116 User-Defined OTD Field Node Properties

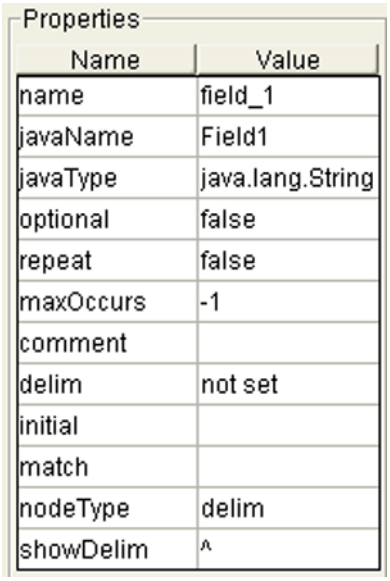


Table 42 Field Node Properties

Name	Description
name	Field display name.
javaName	Property accessor basename, automatically generated—*do not modify*.
javaType	Java type; can be either **java.lang.String** or byte array (**byte[]**).
optional	Specifies whether or not the field can be absent (**true** or **false**). Not applicable if the field is the child of a *choice* node.
repeat	Specifies whether or not the node can appear multiple times (**true** or **false**). Not applicable if the field is the child of a *choice* node.
maxOccurs	Specifies the maximum number of occurrences of the node if the node is repeating. Property has no effect if node is non-repeating, but may show error during validation if set to value >1.
comment	Free-form text (no run-time effect).
delim	Delimiter specification (see **Specifying Delimiters** on page 168).
initial	Initial field value, set when the parent node is created or reset. When provided, it is assigned to the node before the node is populated with any data.
match	If **nodeType** is *delimited* or *fixed*, performs exact match to the data.
nodeType	Specifies the marshal/unmarshal format. The default value is **delim**.
showDelim	Displayed only if **nodeType** is *delim*. Once delimiters are specified, value field shows the delimiters.

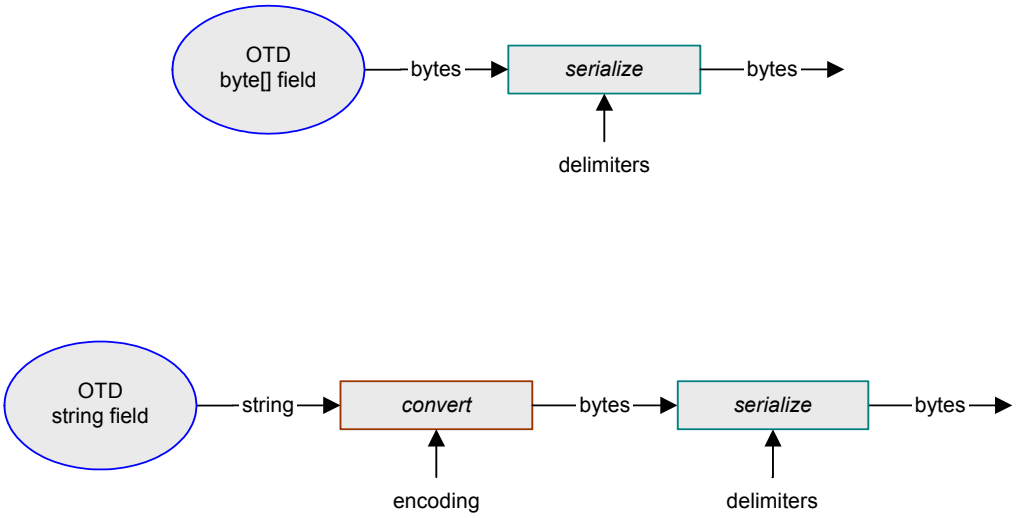
6.5.4 Specifying Data Coding

The data coding you specify when configuring the OTD modifies the Java methods used for marshaling and unmarshaling. The marshaling and unmarshaling processes differ from each other depending upon which Java method you use, and whether you are marshaling to or unmarshaling from byte[] or string fields. The following diagrams attempt to illustrate these differences.

marshal() Method

The simplest marshaling method, **marshal()**, requires *encoding* only when marshaling from a string field, since the parser requires the data in bytes and the output is also in bytes (see Figure 117).

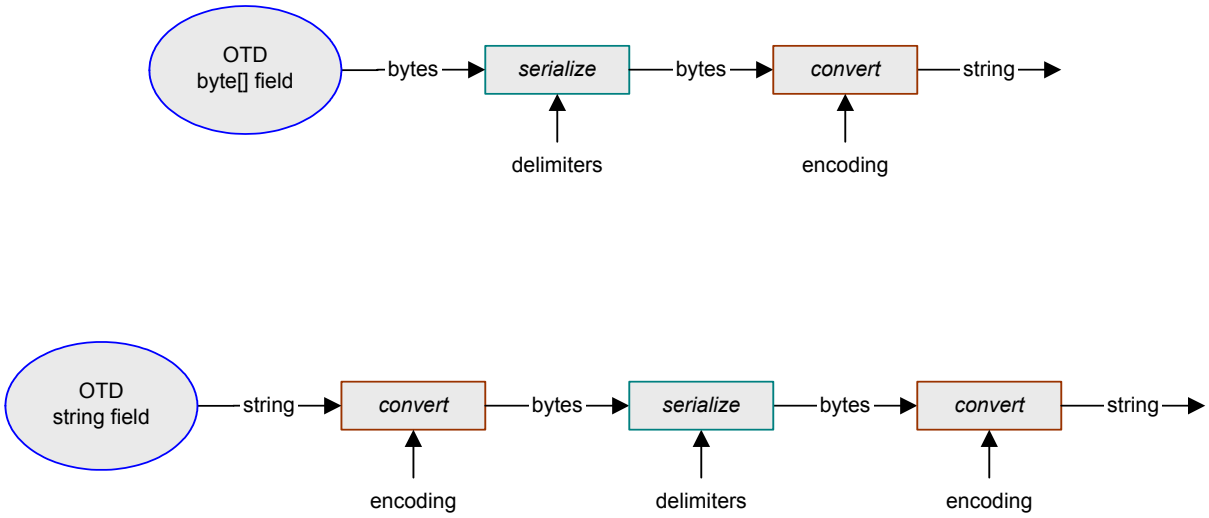
Figure 117 marshal()



marshalToString() Method

The **marshalToString()** method requires *encoding* to produce an output string after marshaling from a byte[] field (see Figure 118). This method also requires *encoding* when marshaling from a string field, since the parser requires the data in bytes, and *encoding* again to produce an output string.

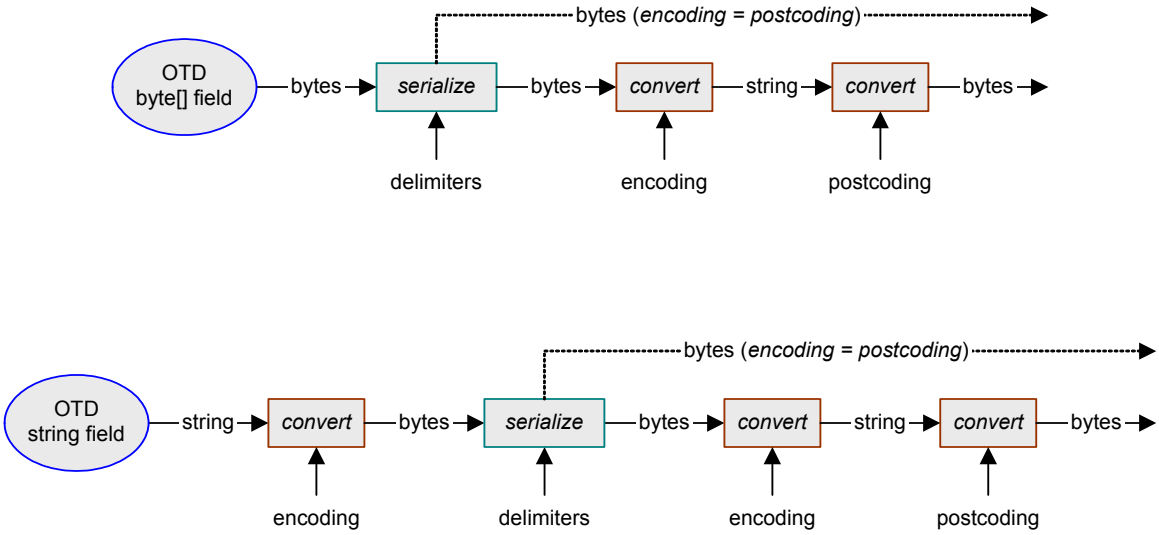
Figure 118 marshalToString()



marshalToBytes() Method

The **marshalToBytes()** method requires *encoding* to produce bytes after marshaling from a string field (see Figure 119). Following serialization, this method also requires *encoding* and *postcoding* to produce an output (in bytes) having a different format from that used by the parser. If the same format is desired, then the *postcoding* property is left undefined, the *encoding* property is substituted by default, and the double conversion is bypassed.

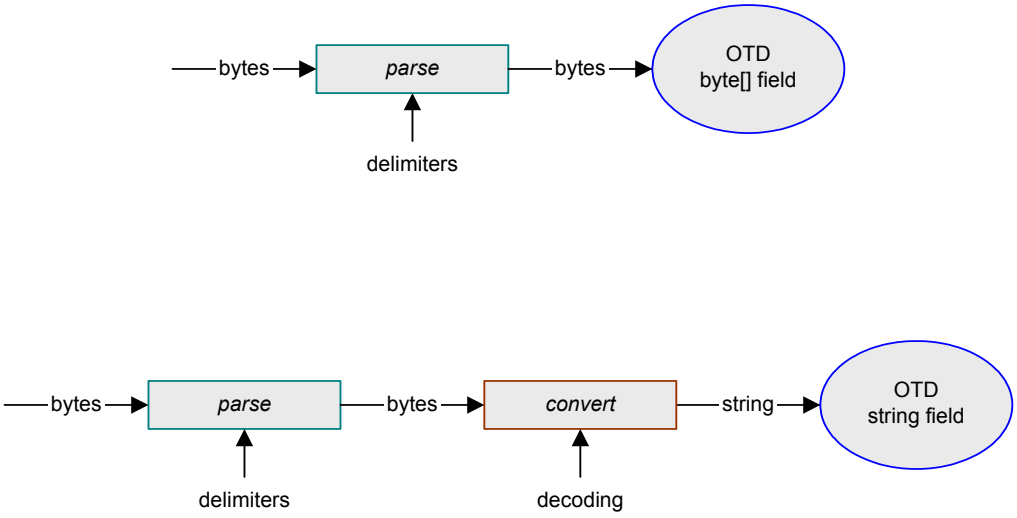
Figure 119 marshalToBytes()



unmarshal() Method

The simplest unmarshaling method, **unmarshal()**, requires *decoding* only when unmarshaling to a string field, since the input is in bytes as required by the parser (see Figure 120).

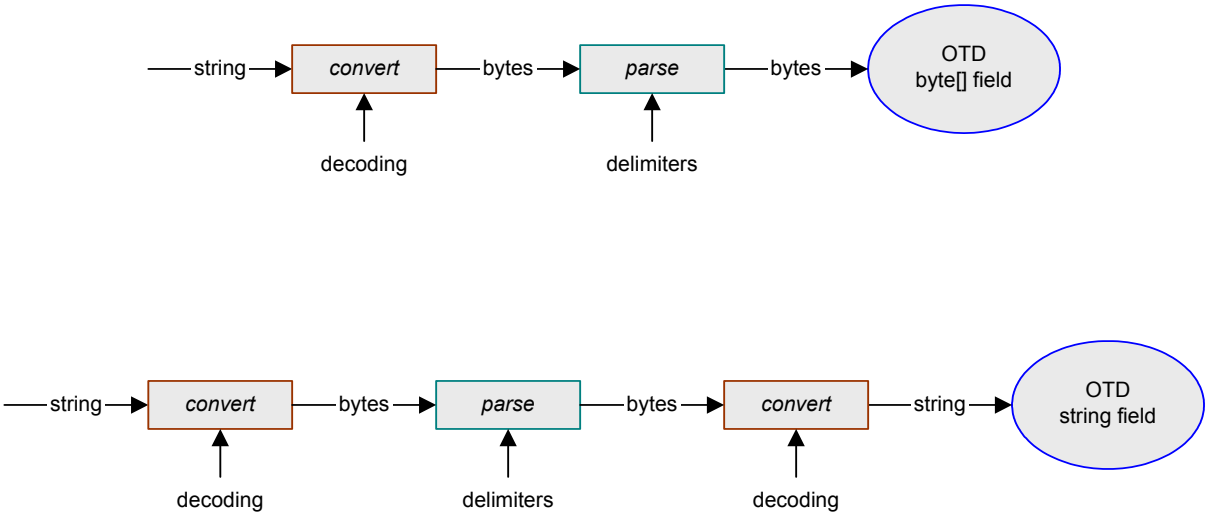
Figure 120 unmarshal()



unmarshalFromString() Method

The **unmarshalFromString()** method requires *decoding* of the input string, since the parser requires the data in bytes (see Figure 121). This method requires a second *decoding* when unmarshaling to a string field.

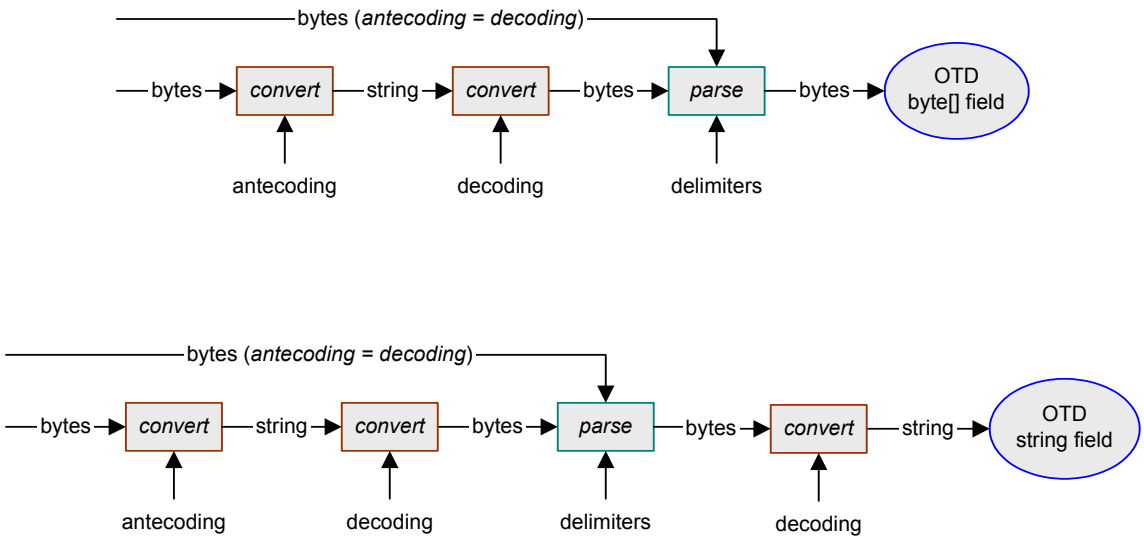
Figure 121 unmarshalFromString()



unmarshalFromBytes() Method

The **unmarshalFromBytes()** method requires *antecoding* and *decoding* if the input data has a different byte format from that used by the parser (see Figure 122). If the same format is desired, then the *antecoding* property is left undefined, the *decoding* property is substituted by default, and the double conversion is bypassed. After parsing, this method requires further *decoding* if unmarshaling to a string field.

Figure 122 unmarshalFromBytes()



6.5.5 Specifying the Node Type

Double-clicking the **nodeType** properties field activates the field for editing. Click the arrow button to display the drop-down menu (see Figure 123 for a root node example). Descriptions of the property options are listed in Table 43.

Figure 123 Node Type Property Options (Root Node Example)

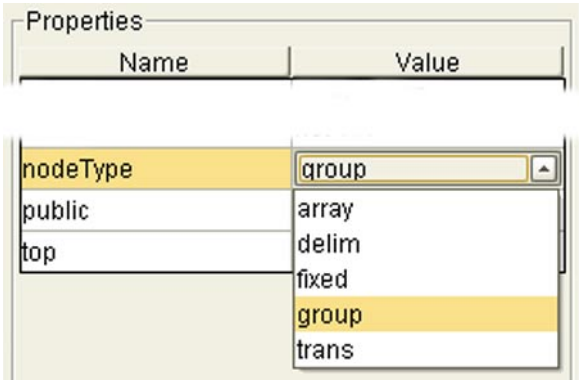


Table 43 Node Type Property Options

Option	Description	Applicable Nodes			
		Root	Element	Field	Internal
alter	Applies only to *element* nodes; option is displayed only if you have checked the *choice node* box when naming the element. Alter (*alternate*) selects one child or another. One, and only one, child is always present after the unmarshal operation.	No	Yes	No	choice
array	Array is a delimited structure. If repeated, occurrences are separated by the *repeat* delimiter. The last occurrence may be terminated by a *normal* delimiter.	Yes	Yes	Yes	simple or group
delim	Delim (*delimited*) structure. If repeated, occurrences are separated by a *normal* delimiter.	Yes	Yes	Yes	simple or group
fixed	Fixed indicates a fixed length, which is specified by non-negative integer (or zero to indicate end of parent node data).	Yes	Yes	Yes	simple or group
group	Group provides organizational grouping for purposes such as repetition. Applies only to non-leaf (*root* and *element*) nodes, not *field* nodes.	Yes	Yes	No	group
trans	Trans (*transient*) appears only in an internal tree as a scratchpad field. It does not appear in external data representation, and can only have *trans* nodeTypes as children.	Yes	Yes	Yes	choice, simple, or group

6.5.6 Specifying Delimiters

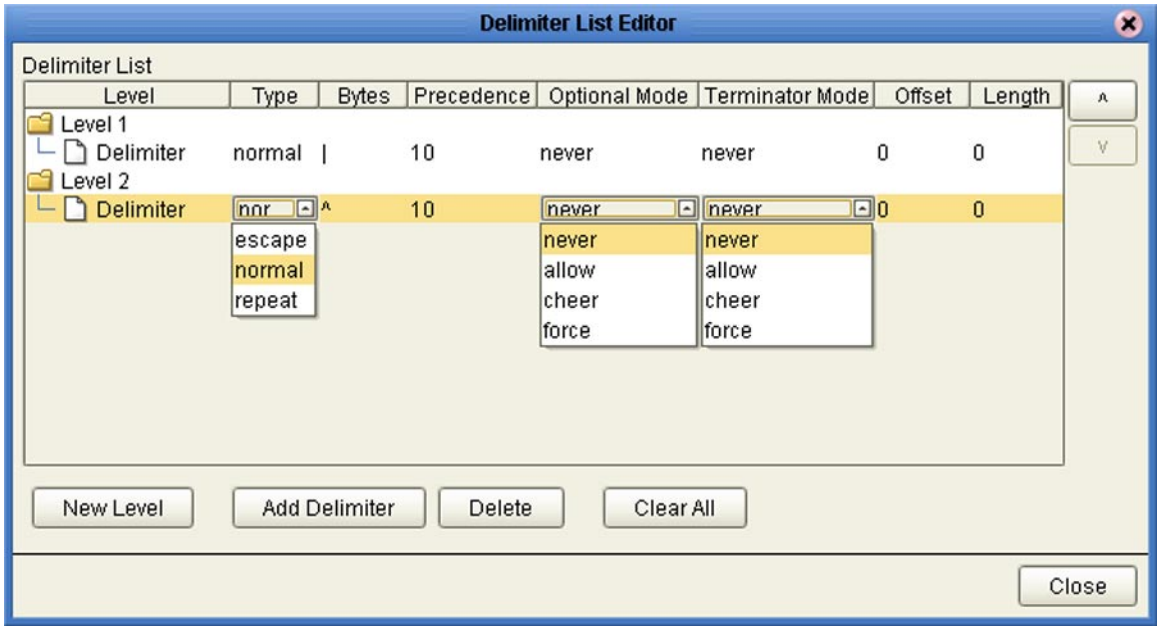
Delimiter List

You can define a set of delimiters—a *delimiter list*—for any node in the hierarchical data structure. This delimiter list is used in the external data representation for that node and its descendents. A delimiter list defined for any node overrides the effect of any ancestor node's delimiter list on the node and its descendents.

Delimiters are defined using the Delimiter List Editor (see Figure 124). The editor is invoked by clicking in the **delim** property *value* field and clicking the ellipsis (**...**) button, or by double-clicking in the field. See **Creating a Delimiter List** on page 178.

Clicking within a field in the Delimiter List Editor enables the field for editing. Double-clicking within one of the three fields illustrated in Figure 124 displays its drop-down menu.

Figure 124 Delimiter List Editor



The command buttons in the Delimiter List Editor are enabled only when the action is allowed for the selected line item.

Table 44 Delimiter List Editor Command Buttons

Command	Action
New Level	Adds a new level after the selected level.
Add Delimiter	Adds a new delimiter after the selected delimiter, or to the bottom of list under the selected level.
Delete	Deletes the selected line item (level or delimiter) from the list.
Clear All	Deletes all levels and delimiters from the list, following confirmation (displays a warning box first).
Move Up (^)	Moves the selected level up one level; moves the selected delimiter up one step within the same level or to the bottom of the previous level.
Move Down (v)	Moves the selected level down one level; moves the selected delimiter down one step within the same level, or to the top of the next level (if there is one).
Close	Closes the editor and saves your entries.

You can also right-click anywhere within the table area to invoke a popup menu (see Figure 125), which contains options duplicating the actions of the command buttons in the editor.

Figure 125 Popup Menu

Delimiter Properties

Table 45 Delimiter Properties and Value Options

Property	Option	Description
Level		Delimiter levels, assigned consecutively to delimited nodes in the OTD node hierarchy.
Type	escape	Escape sequence.
	repeat	Array delimiter/separator.
	normal	Terminator.
Bytes		Delimiter character(s). **Note:** Entering a *Length* or *Offset* clears this field.
Precedence		See **Precedence** on page 175.
Optional Mode	never	Do not allow on input, do not emit on output (empty field between delimiters implies zero length data field).
	allow	Skip empty field if present; if absent, do not delimit on output.
	cheer	Skip empty field if present; if absent, delimit on output.
	force	Require empty, delimited field on input; always delimit on output. Only the *force* option allows trailing delimiters for a sequence of absent optional nodes.
Terminator Mode	never	Do not allow on input, do not emit on output (pure separator).
	allow	Allow on input, do not emit on output.
	cheer	Allow on input, always emit on output.
	force	Require on input, always emit on output (pure terminator).
Offset		Delimiter offset (non-negative integer). You can specify the delimiter by length and offset of the data field, rather than delimiter *Byte*s.
Length		Delimiter length (positive integer). You can specify the delimiter by length and offset of the data field, rather than delimiter *Byte*s.

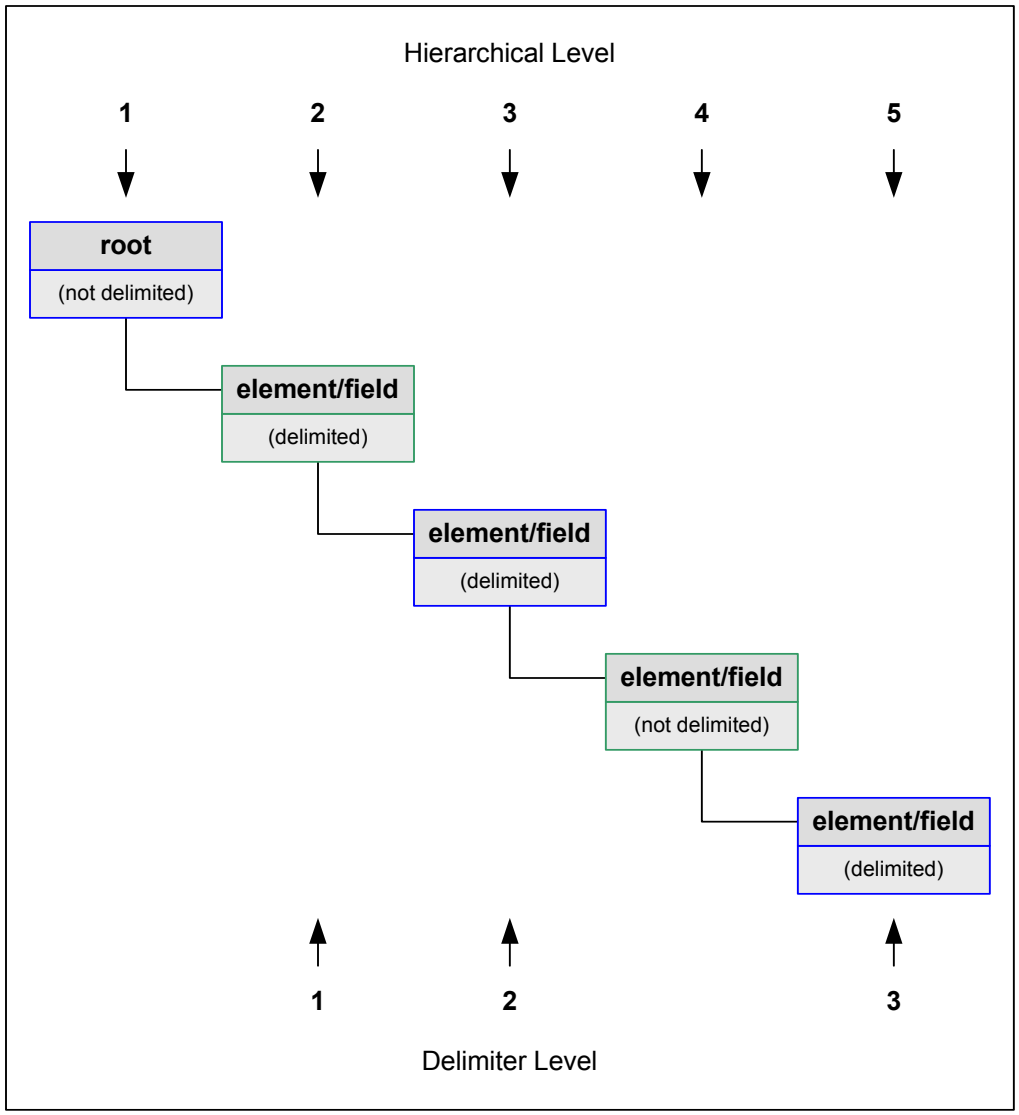
Note: *You should avoid using a colon (:) as a delimiter character, since it is used as a literal in system-generated time strings. This can interfere with recovery procedures following a Logical Host shutdown, for example.*

Delimiter Levels

Delimiter levels are assigned in order to those hierarchical levels of an OTD that contain at least one node that is specified as being delimited. If none of the nodes at a particular hierarchical level is delimited, that hierarchical level is skipped in assigning delimiter levels.

Delimiter lists are typically specified on the root node, so that the list applies to the entire OTD. The root node itself is typically not delimited, so that *Level 1* applies to those nodes that are children of the root node. See Figure 126 and the following example.

Figure 126 OTD Hierarchical and Delimiter Levels



For example, if you want to parse the following data:

```
a^b|c^d|e
```

you might create a User-Defined OTD as follows:

- root
  - element1
    - field1
    - field2
  - element2
    - field3
    - field4
  - field5

In this example, the delimiter list is specified on the *root* node, which is not delimited; therefore, the list has two levels:

- Level 1
  - Delimiter |
- Level 2
  - Delimiter ^

The *Level 1* delimiter (|) applies to element1, element2, and field5. The *Level 2* delimiter (^) applies to field1 - field4.

If the root node is set to be delimited, the *Level 1* delimiters will then apply to it. Using the above example, the *Level 2* delimiter (^) would then apply to element1, element2, and field5, and a new *Level 3* delimiter would apply to field1 - field4.

Delimiter lists can be much more complex than this very simple example. For instance, you can create multiple delimiters of different types at any given level, and you can specify a delimiter list on any node within the OTD—not only the root node as shown in the example. See **Creating a Delimiter List** on page 178 for a step-by-step description of the procedure for creating a Delimiter List.

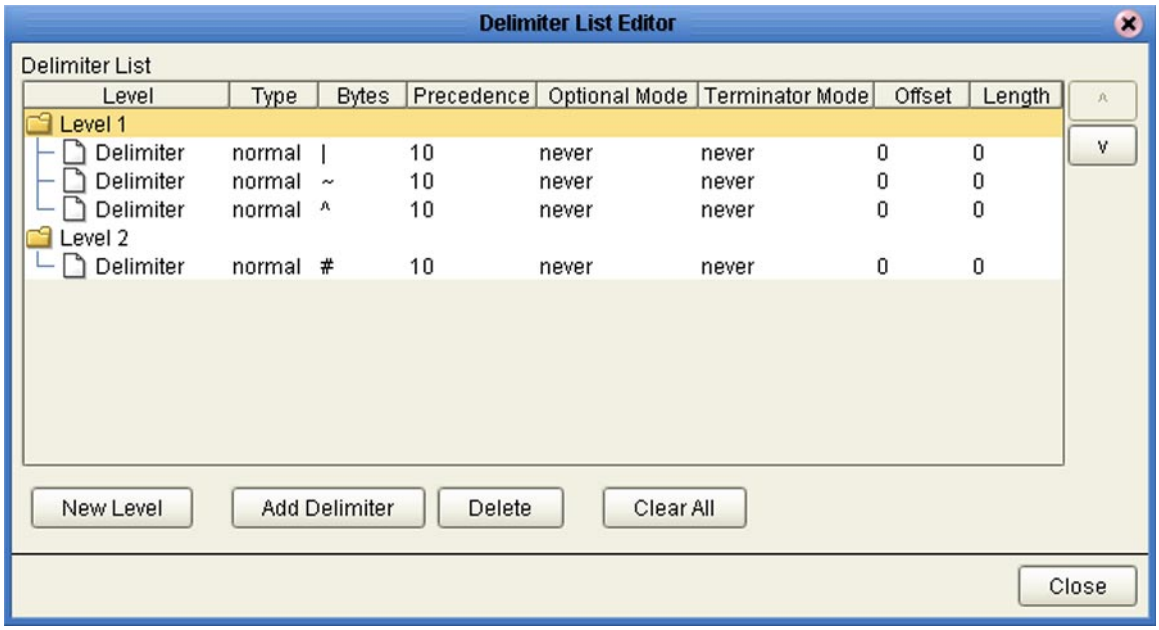
Multiple Delimiters

You can specify multiple delimiters at a given level; for example, if you specify |, ~, and ^ as delimiters for a specific level (see Figure 127), the parser will accept any of these delimiters:

- root
  - element (delimiters = "|", "~", "^")
    - field1 (delimiter = "#")
  - field2 (delimiters = "|", "~", "^")

This will successfully parse the data **abc|def**, **abc~def**, and **abc^def**.

Figure 127 Multiple Delimiter Example



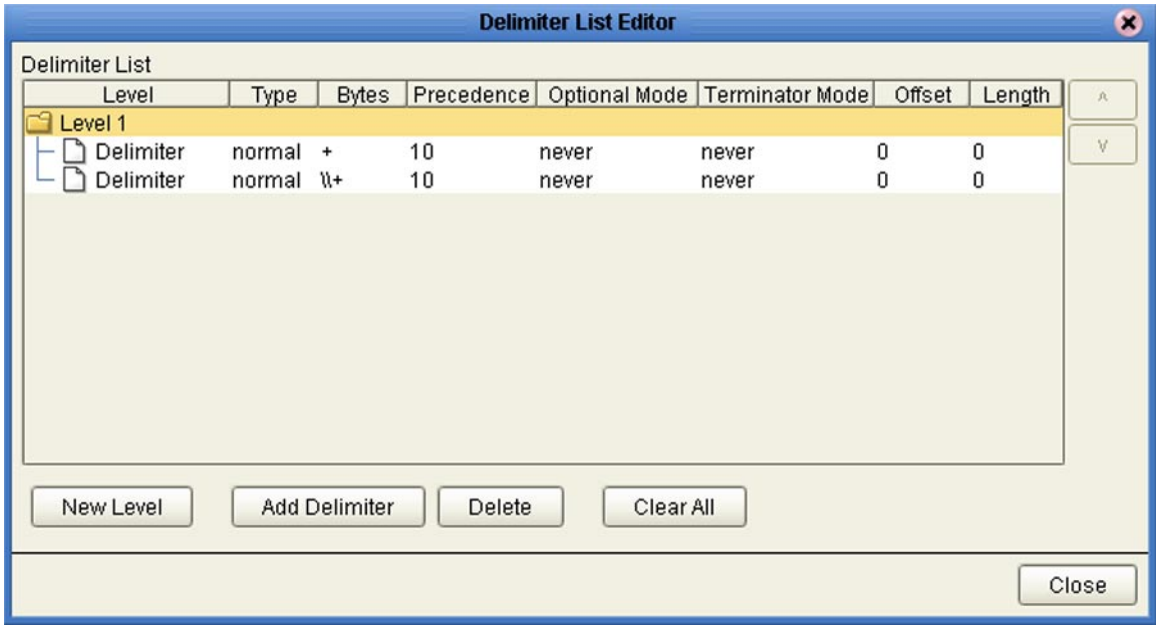
Delimiter Type

Escape Option

An *escape* delimiter is simply a sequence that is recognized *and ignored* during parsing. Its purpose is to allow the use of escape sequences to embed byte sequences in data that would otherwise be seen as delimiter occurrences.

For example, if there is a normal delimiter "**+**" at a given level, and we define an escape delimiter "**\+**" (see Figure 128), then **aaa+b\+c+ddd** will parse as three fields: **aaa**, **b\+c**, and **ddd**. If the escape delimiter were not defined, the sequence would then parse as four fields: **aaa**, **b**, **c**, and **ddd**.

Figure 128 Escape Type Delim



If there is *only* an escape delimiter on a given level, however, it presents a *no delimiter defined* situation for **delim** and **array** nodes.

Precedence

Precedence (see **Figure 138 on page 180**) indicates the priority of a certain delimiter, relative to the other delimiters. By default, all delimiters are at precedence 10, which means they are all considered the same; fixed fields are hard-coded at precedence 10. Delimiters on parent nodes are not considered when parsing the child fields; only the child's delimiter (or if it is a fixed field, its length). The range of valid precedence values is from 1 to 100, inclusive.

Changing the precedence of a delimiter will cause them to be applied to the input data-stream in different ways. For example:

- root
  - element (type delim, delimiter = "^", repeat)
    - field1 (type fixed, length = 5)
    - field2 (type fixed, length = 8, optional)

Although this will parse **'abcde12345678^zyxvuABCDEFGH'**, it will *not* parse the text **'abcde^zyxvuABCDEFGH'** even though the second fixed field is optional. The reason is that the element's delimiter is ignored within the fixed field because they have the same precedence. If you want the element's delimiter to be examined within the fixed field data, you must change its precedence, for example:

- root
  - element (type delim, delimiter = "^", repeat, **precedence = 11**)
    - field1 (type fixed, length = 5)
    - field2 (type fixed, length = 8, optional)

This will successfully parse the text **'abcde^zyxvuABCDEFGH'**.

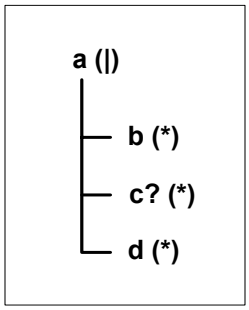
A similar argument can be applied to delimited child nodes. The parser normally attempts to match the child delimiter—setting the precedence to 11 forces the parser to match the parent delimiter first.

Optional Mode

Consider the tree structures shown in Figure 129 and Figure 130, where the node **a** has a pipe (|) as its delimiter, and the sub-nodes **b**, **c**, and **d** all have asterisks (*) as their delimiters.

- **Example 1:** Sub-node **c** is *optional*. (Sub-node **c** and sub-node **d** must have different values for the *match* parameter.)

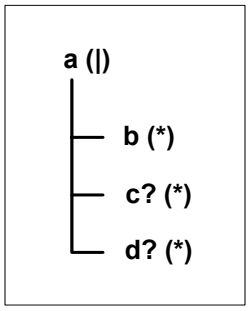
Figure 129 Optional Mode Property (Example 1)



Option	Input	Output
never	**b*d|**	**b*d|**
allow	**b**d|**	**b*d|**
cheer	**b**d|**	**b**d|**
force	**b**d|**	**b**d|**

- **Example 2:** Both sub-node **c** and sub-node **d** are *optional*.

Figure 130 Optional Mode Property (Example 2)

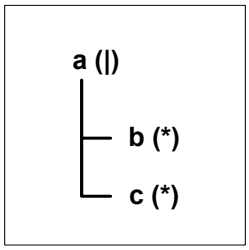


Option	Input	Output
never	**b|**	**b|**
allow	**b|**, **b*|**, or **b**|**	**b|**
cheer	**b|**, **b*|**, or **b**|**	**b**|**
force	**b**|**	**b**|**

Terminator Mode

Consider the tree structure shown in Figure 131, where the node **a** has a pipe (|) as its delimiter, the sub-nodes **b** and **c** have asterisks (*) as their delimiters.

Figure 131 Terminal Mode Property Example



Option	Input	Output
never	**c**|	**c**|
allow	**c**| or **c***|	**c**|
cheer	**c**| or **c***|	**c***|
force	**c***|	**c***|

Escape Sequences

The following escape sequences are supported in delimiters.

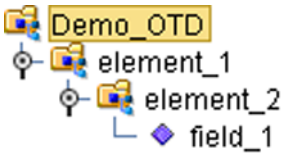
Table 46 Escape Sequences

Sequence	Description
\\	Backslash
\b	Backspace
\f	Linefeed
\n	Newline
\r	Carriage return
\t	Tab
\ddd	Octal number
\xdd	Hexadecimal number

6.5.7 Creating a Delimiter List

As an example, we shall create a delimiter list for the simple OTD structure shown in Figure 132.

Figure 132 Demo OTD Tree



To create a delimiter list

1 In the OTD Editor, select the node for which you want to define a set of delimiters (this example uses the *root* node). Initially, the value for the **delim** property appears as *not set*, as shown in Figure 133.

Figure 133 Initial Root Node Properties

Properties

Name	Value
name	Demo_OTD
javaName	Demo_OTD
comment	
delim	not set
nodeType	group
public	false
top	true

2 Double-click in the **delim** value field to activate it for editing; an ellipsis (**...**) button appears, as shown in Figure 134. The **showDelim** property fields also appear.

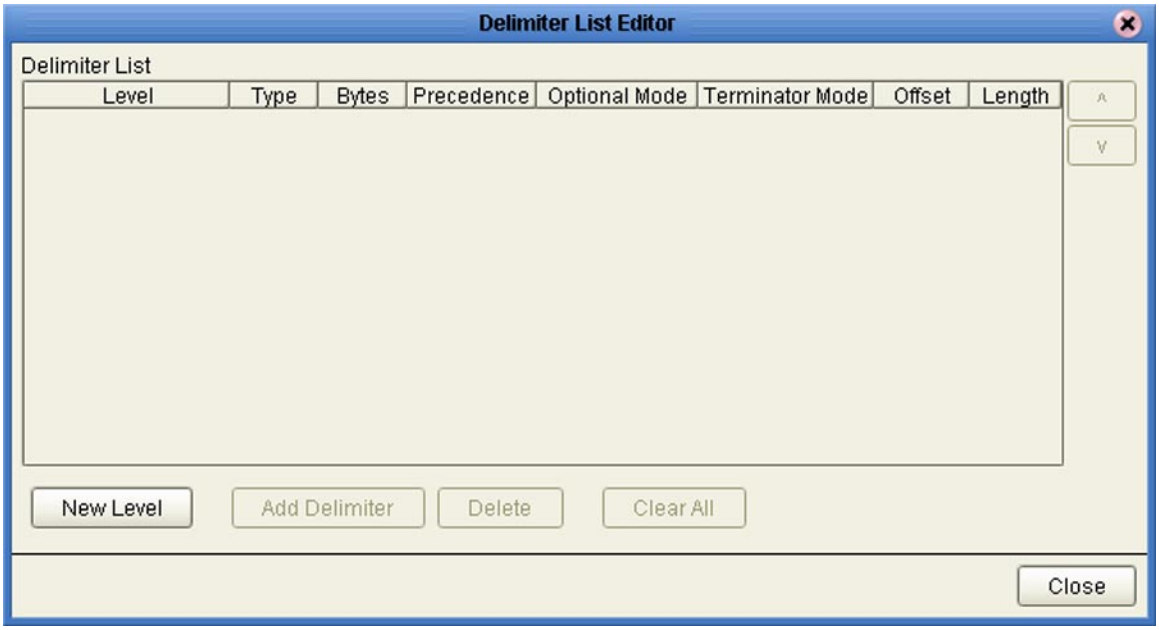
Figure 134 Activated *delim* Value Field

Properties

Name	Value
name	Demo_OTD
javaName	Demo_OTD
comment	
delim	not set ...
nodeType	group
public	false
top	true

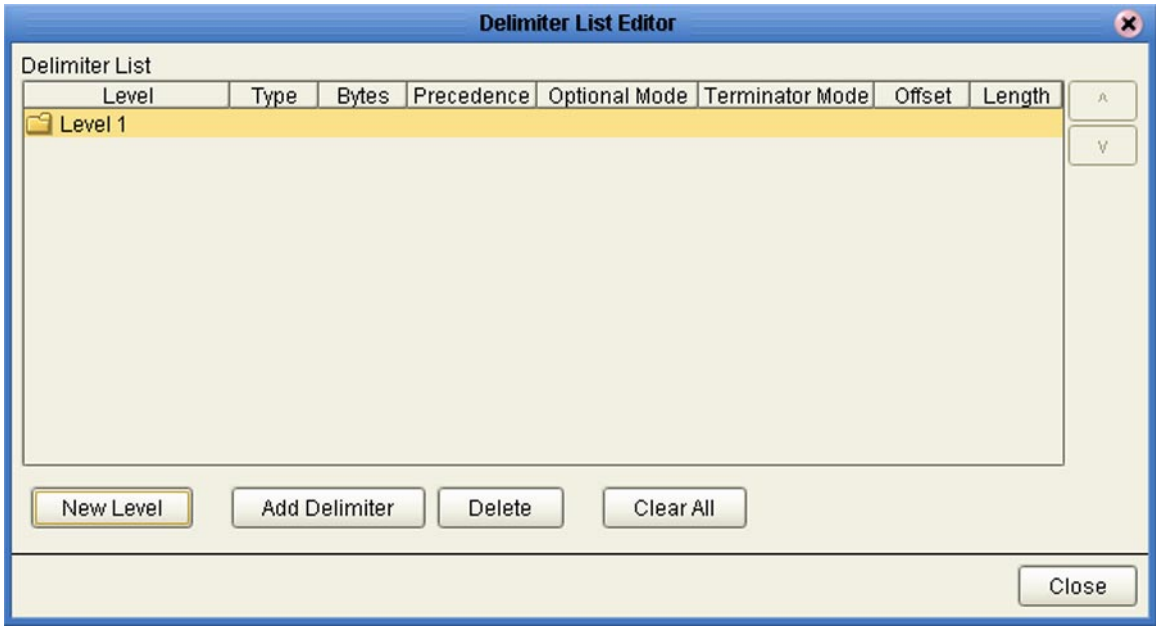
3 Click the ellipsis button to display the Delimiter List Editor, which is initially blank (see Figure 135).

Figure 135 Delimiter List Editor - Delimiters Not Set



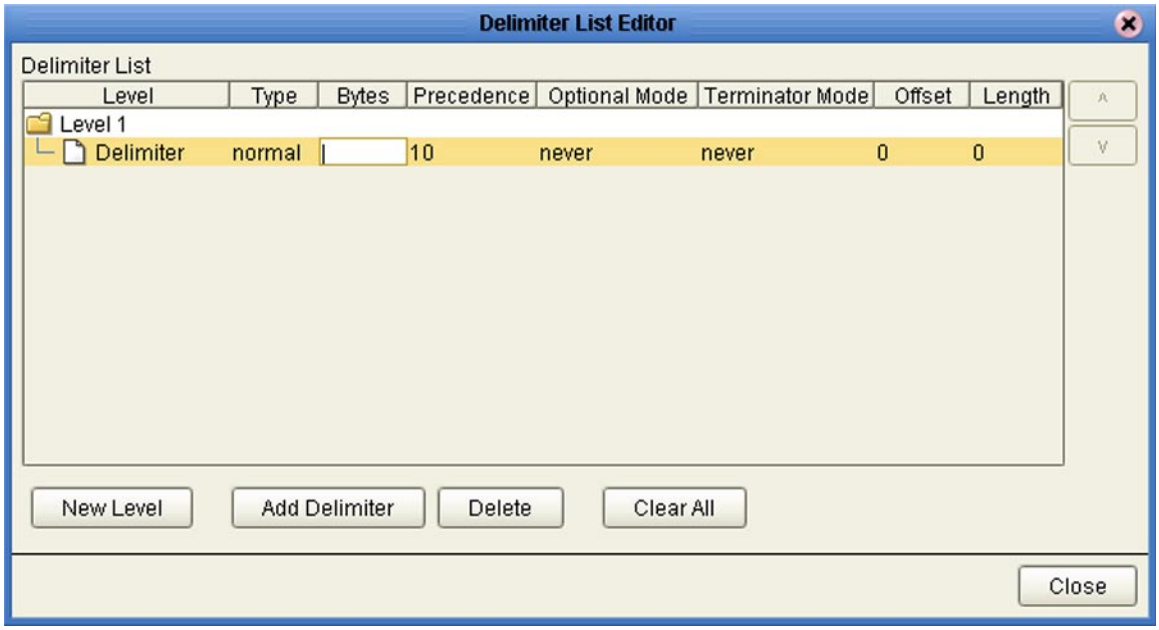
4 Click **New Level** to add a level to the delimiter list, as shown in Figure 136.

Figure 136 Delimiter List Editor - Insert Level



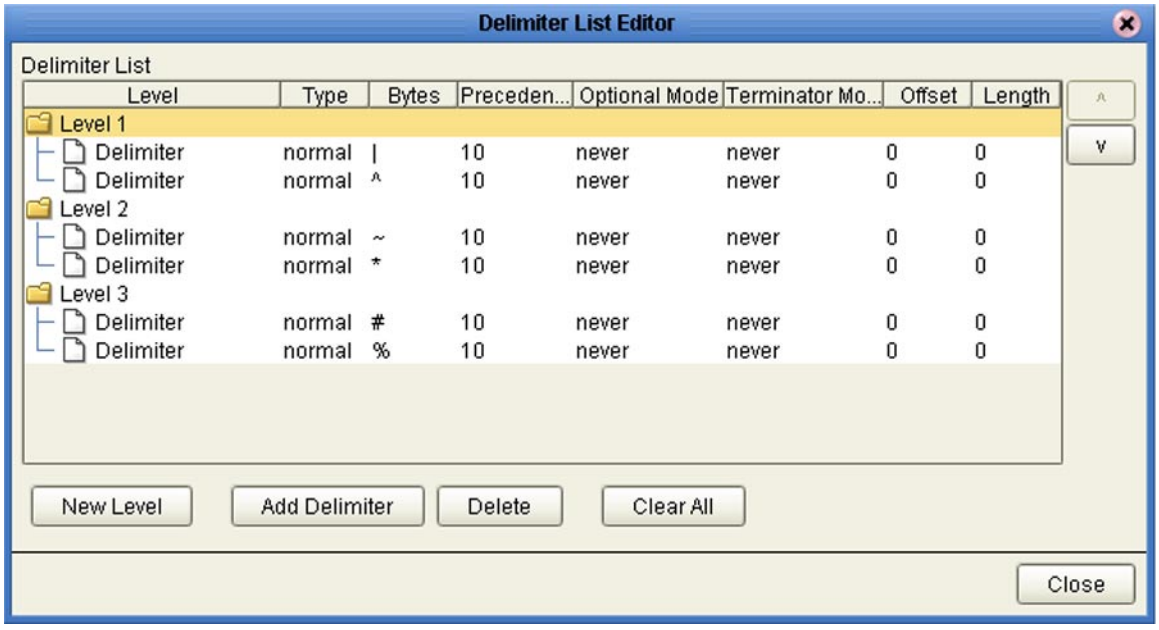
5 Select a level and click **Add Delimiter** to add a delimiter to the selected level. Double-click in the Bytes field and type in the delimiter characters (see Figure 137).

Figure 137 Delimiter List Editor - Add Delimiter



6 Continue adding levels and delimiters as required (see Figure 138).

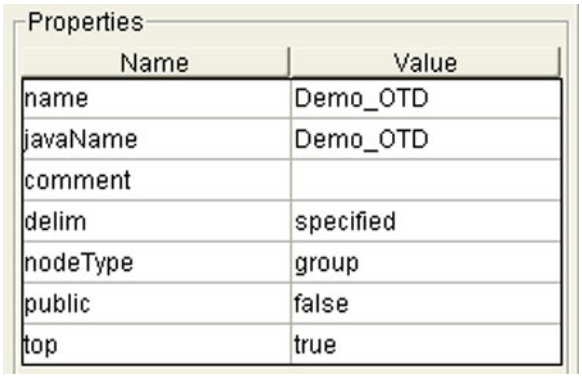
Figure 138 Delimiter List Editor - Add Levels and Delimiters



7 Click **Close** to close the editor and save your work.

8 The value for the **delim** property will now appear as *specified*, as shown in Figure 139.

Figure 139 Root Node - Delimiter Specified

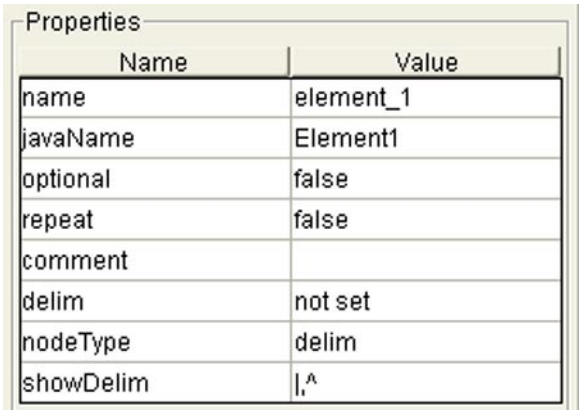

Properties

Name	Value
name	Demo_OTD
javaName	Demo_OTD
comment	
delim	specified
nodeType	group
public	false
top	true

9 Since the **nodeType** values for elements and fields are *delim* by default, they will automatically display the delimiters for their respective levels. To exclude any of them from the effect of the delimiter list, you must reset their **nodeType** properties to a value other than *delim*.

A The delimiters for *Element_1* are now displayed as shown in Figure 140—compare the delimiters with the list shown in Figure 138. Note that the **delim** property remains *not set*, since the delimiter list is defined for the *root* node. Defining another delimiter list here will override the original list.

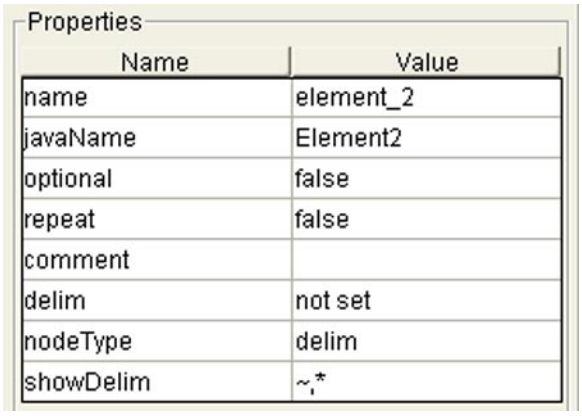
Figure 140 Element_1 Node Properties


Properties

Name	Value
name	element_1
javaName	Element1
optional	false
repeat	false
comment	
delim	not set
nodeType	delim
showDelim	|,^

B The delimiters for *Element_2* are now displayed as shown in Figure 141—compare the delimiters with the list shown in Figure 138. Again, the **delim** property remains *not set*.

Figure 141 Element_2 Node Properties



C The delimiters for *Field_1* are now displayed as shown in Figure 142— compare the delimiters with the list shown in Figure 138. Again, the **delim** property remains *not set*.

Figure 142 Field_1 Node Properties

10 Once you have defined your delimiter list, you should test the OTD to verify that it parses correctly (see **Using the OTD Tester** on page 132).

6.6 OTD Libraries

Several libraries containing large numbers of OTDs for specific purposes are available to use with eGate Integrator. These OTDs provide templates corresponding to standard message types defined by various organizations. The templates are pre-defined and can be used as-is or modified using the OTD Editor, whichever is appropriate to your application.

6.6.1 CBO OTD Library

The Canonical Business Object (CBO) OTD Library corresponds to the Open Applications Group's Integration Specification (OAGIS), and is packaged with eGate Integrator. The library is contained in the archive file **CBO_OTD_v1_0.sar**, which can be downloaded from the installation CD-ROM.

OAGIS provides a horizontal business language that provides an open interface for interaction with vertical industry standards. Extensive information is available from the Open Applications Group's Web site **http://www.openapplications.org.**

6.6.2 Add-on OTD Libraries

Several other OTD libraries are available from SeeBeyond as add-ons to eGate Integrator, including libraries for SWIFT and HL7.

Chapter 7

Collaboration Definitions (Java)

This chapter describes the features of the Collaboration Definition Editor (Java) and outlines procedures for building Java-based Collaboration Definitions.

What's in This Chapter

7.1 Overview

Collaboration Definitions define how data should be processed and routed between Project components, how databases should be queried in response to requests, and how APIs to one or more applications should be invoked. The external data formats that characterize the input and output data structures in a Collaboration Definition are described by Object Type Definitions (OTDs).

A Collaboration will typically receive a message containing the external representation of a particular OTD. It will use the *unmarshal* method of an instance of that OTD to parse the data and make it accessible though the hierarchical data structure. Then it will perform some operation—for example, copying parts of the data to another OTD instance. Finally, it will invoke the *marshal* method on the other OTD instance to render the contents of its data structure as a single, serialized data stream for further transport.

At run time, an OTD instance is accessed directly from Java, using accessors resembling Java beans. Each of the nodes comprising the hierarchy of the data structure has a set of properties with *get* and *set* methods. See **Object Type Definitions** on page 119 for additional information.

Important: *If you delete an OTD in the Project Explorer, any Collaboration Definitions that have been built using that OTD will be affected. It is recommended that you run the Impact Analyzer before attempting to delete any OTDs (see* **Impact Analyzer** *on page 77).*

As with other eGate Integrator components, it is essential to manage versions of Collaboration Definitions carefully. See **Version Control** on page 79 for descriptions of various version control features applicable to Collaboration Definitions.

The Enterprise Designer includes two primary tools, the Collaboration Definition Wizard (Java) and Collaboration Editor (Java), that you use to create and customize your Java-based Collaboration Definitions. These tools are described in the following sections.

7.2 Using the Collaboration Definition Wizard (Java)

The Collaboration Definition Wizard (Java) guides you through the initial phases of creating a Java-based Collaboration Definition, and then invokes the Collaboration Editor (Java). The user interface is highly self-explanatory, but details of the navigation buttons are listed in Table 47 for your reference.

Table 47 Wizard Navigation Buttons

Button	Function
< Back	Returns to the previous step in the wizard. This button is disabled on the first step.
Next >	Goes to the next step in the wizard. This button is disabled on the last step.
Finish	Saves all Collaboration Definition settings and closes the wizard. This button is only enabled on the last step.
Cancel	Closes the wizard without saving the Collaboration Definition.
Help	Displays the online help documentation for the Collaboration Definition Wizard dialog box.
Add	Adds a selected Object Type Definition to the Collaboration Definition.
Remove	Removes a selected Object Type Definition from the Collaboration Definition.

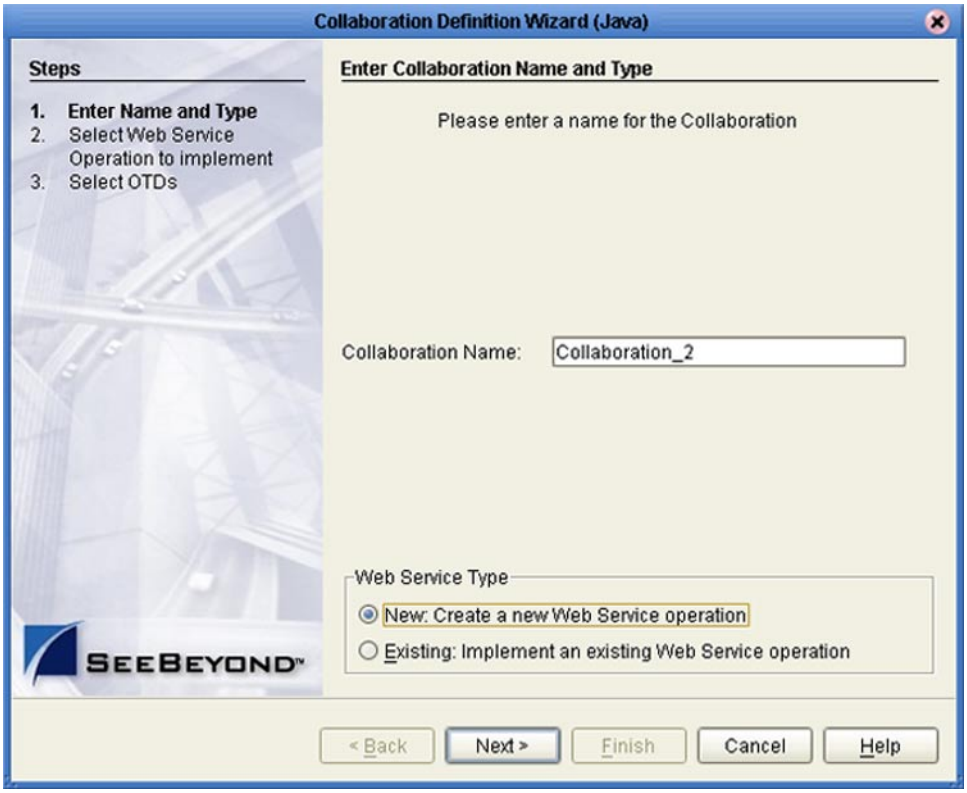
7.2.1 Creating a Java-based Collaboration Definition

Note: *Any changes made to the names of the Collaborations should be done when the Collaboration is created. If the name is changed later, the Collaboration should be opened, regenerated if applicable, and saved again. To be safe, this should also be done before creating the Connectivity Map and Deployment Profile.*

To create a Java-based Collaboration Definition

1. Right-click on a Project in the Enterprise Explorer to display the Project context menu.
2. Select **New > Collaboration Definition (Java)** to invoke the Collaboration Definition Wizard (Java).
3. Enter a **Name** for your Collaboration, as shown in Figure 143.

Figure 143 Initial Wizard Dialog



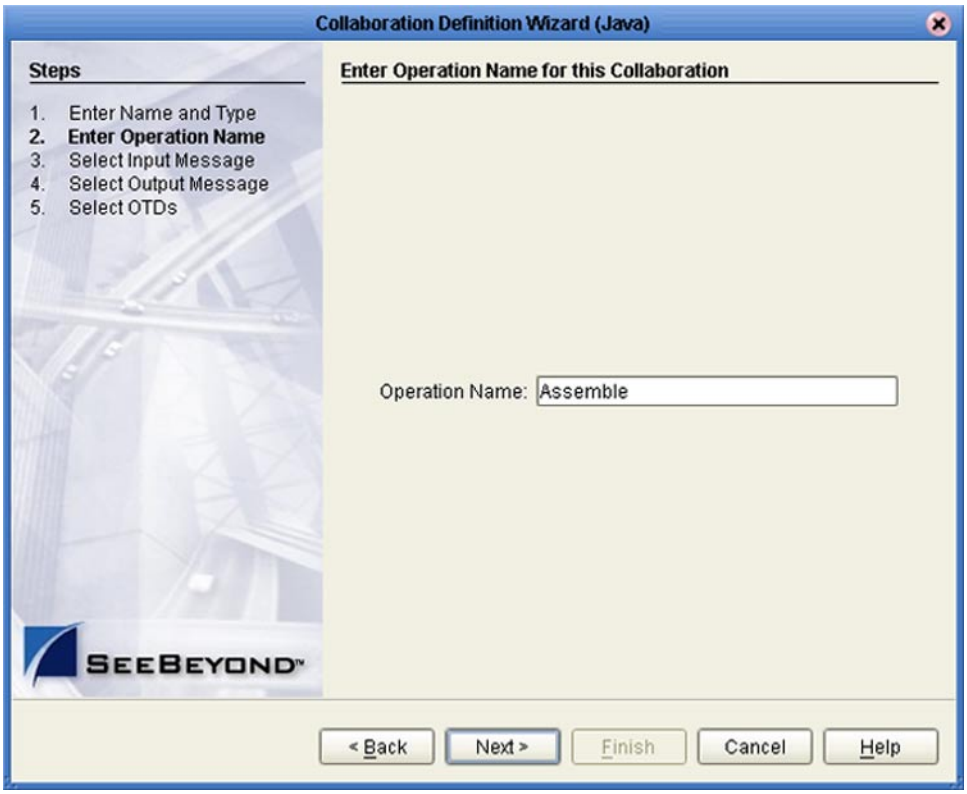
4. Select a Web Service Type, which can be either:
   - A New Web Service.
   - An Existing Web Service (for example, an eInsight process or an OTD).
5. Click **Next** to proceed to the next Wizard dialog, which is dependent upon your Web Service Type selection.

New Web Service

If you selected a New Web Service, you will be presented with the following set of Wizard dialogs.

1 Enter an operation name, as shown in Figure 144. This will become the *method* that can be used to invoke the Java-based Collaboration as a Web service.

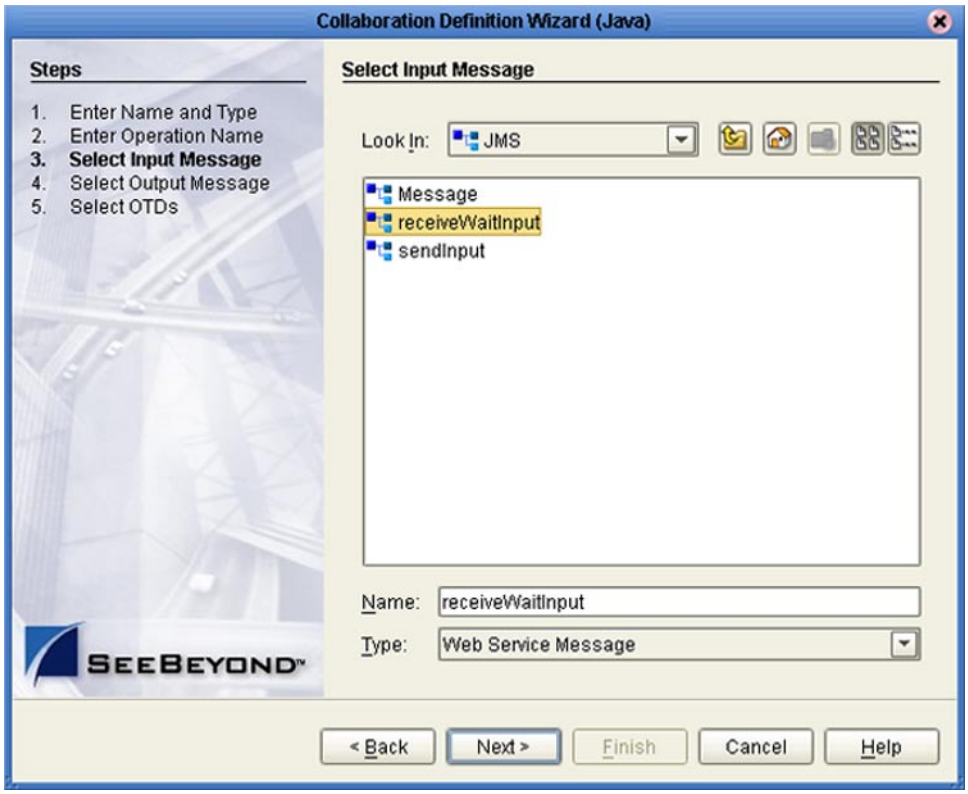
Figure 144 New Web Service: Operation Name



2 Click **Next** to proceed to the next Wizard dialog.

3 Select the input Web service message, as shown in Figure 145.

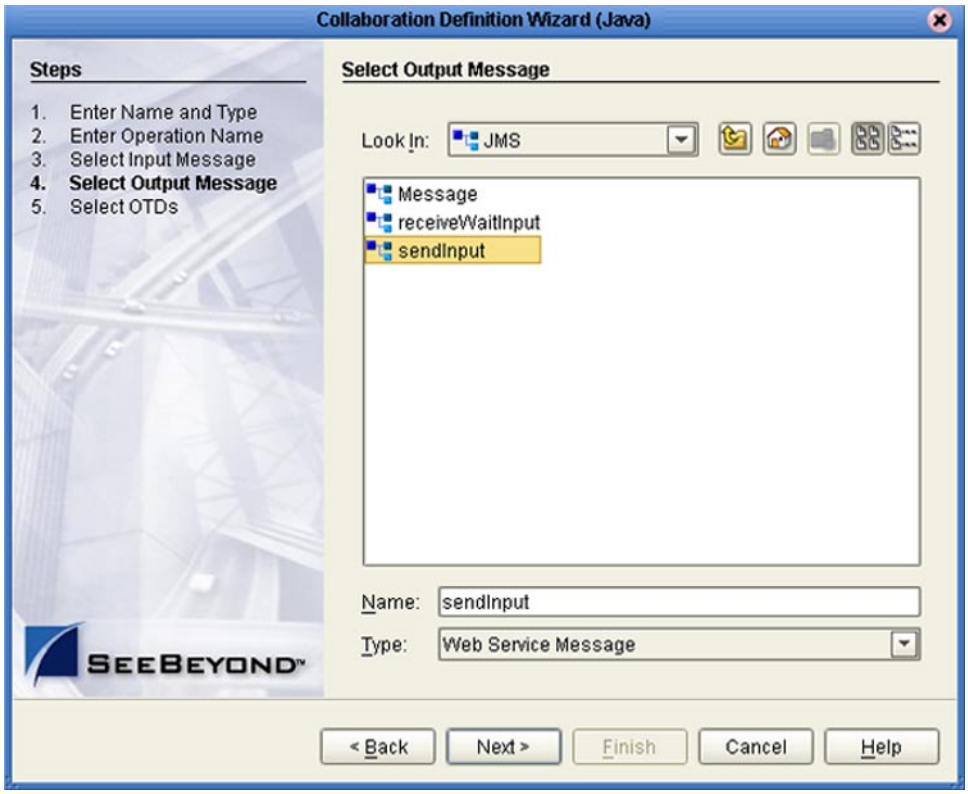
Figure 145 New Web Service: Input Message



4 Click **Next** to proceed to the next Wizard dialog.

5 Select the output Web service message, as shown in Figure 146.

Figure 146 New Web Service: Output Message

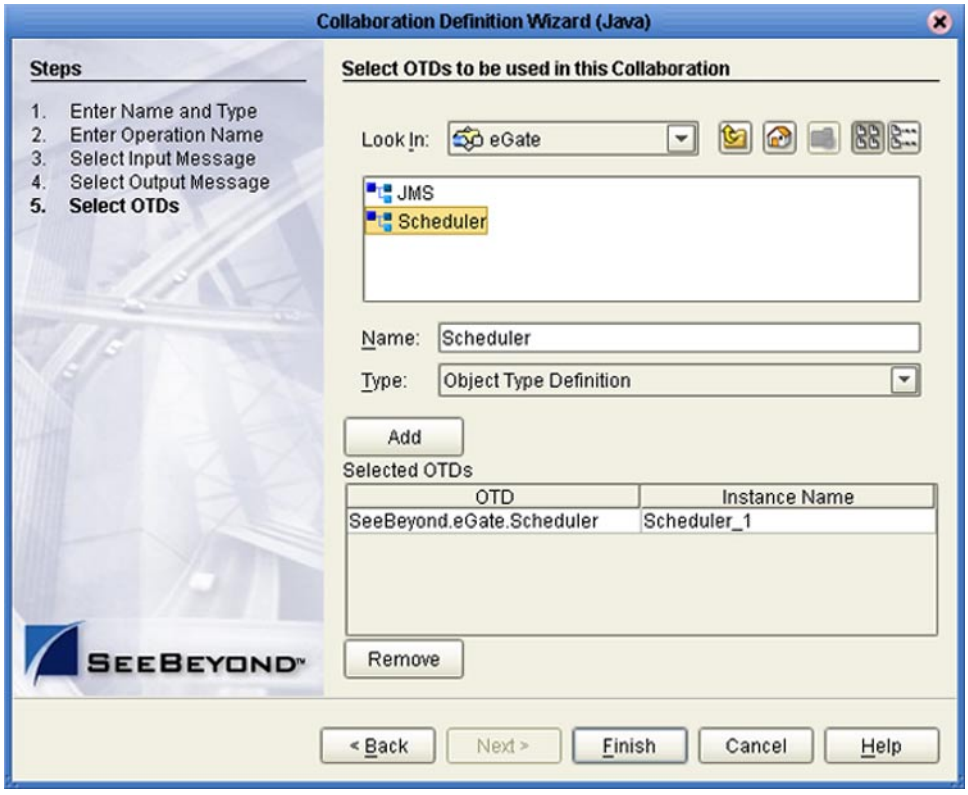


6 Click **Next** to proceed to the next Wizard dialog.

7 Select an auxiliary OTD, as shown in Figure 147. This step is optional, and is intended to support additional functionality such as a database lookup.

Note: *Use caution here, since you may already have OTDs selected in the preceding steps.*

Figure 147 New Web Service: Auxiliary OTD



8 Click **Add** to add the OTD to the Collaboration Definition.

9 Click **Next** to proceed to the Collaboration Editor (Java).

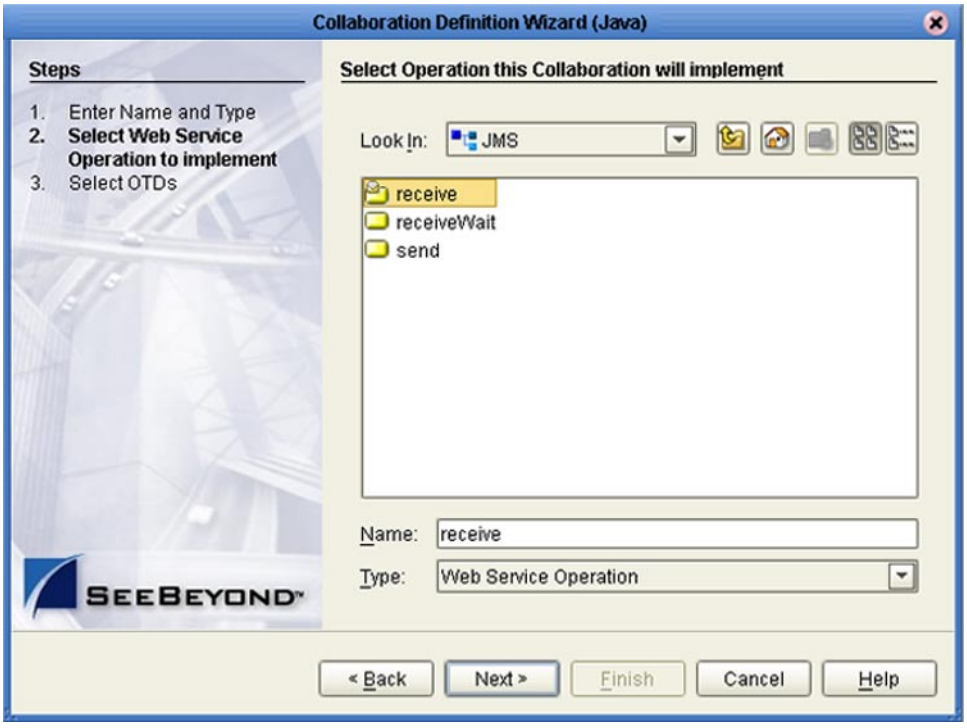
Existing Web Service

If you selected an Existing Web Service, you will be presented with the following set of Wizard dialogs.

1 Select a Web service operation, which can be either:

- An installed ICAN Web Service (for example, a JMS *receive* Web service - see Figure 148).
- A custom Web Service (for example, something that has been created in an ICAN Project).

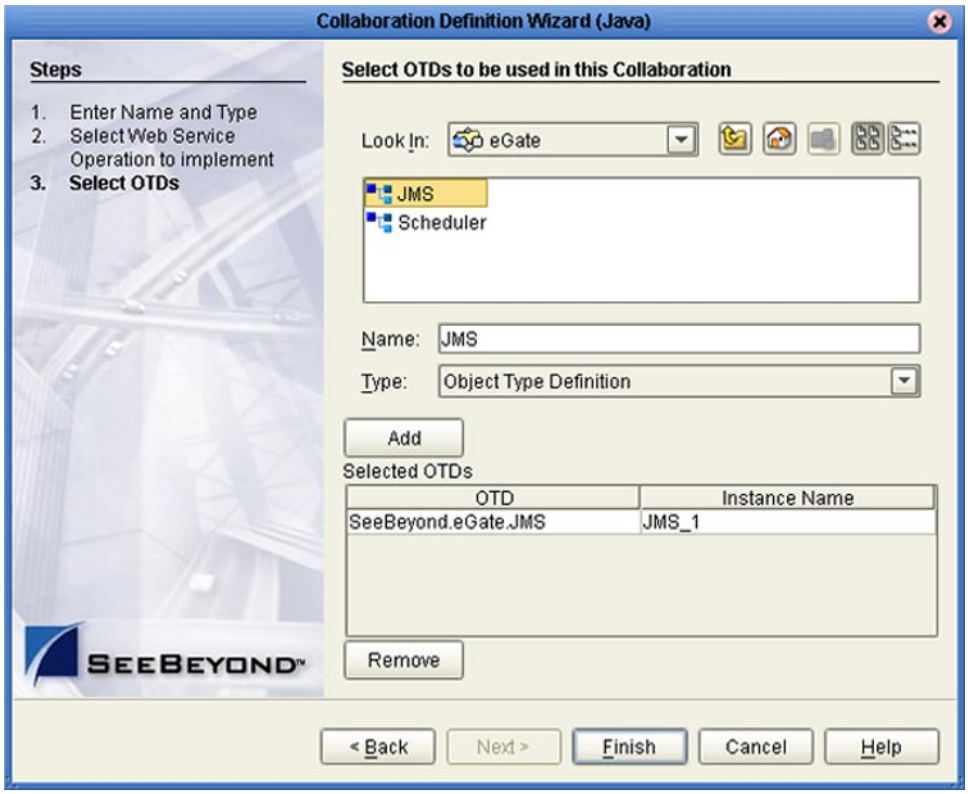
Figure 148 Existing Web Service: Select Operation



2 Click **Next** to proceed to the next Wizard dialog.

3 Select an OTD, as shown in Figure 149.

Figure 149 Existing Web Service: Select OTD



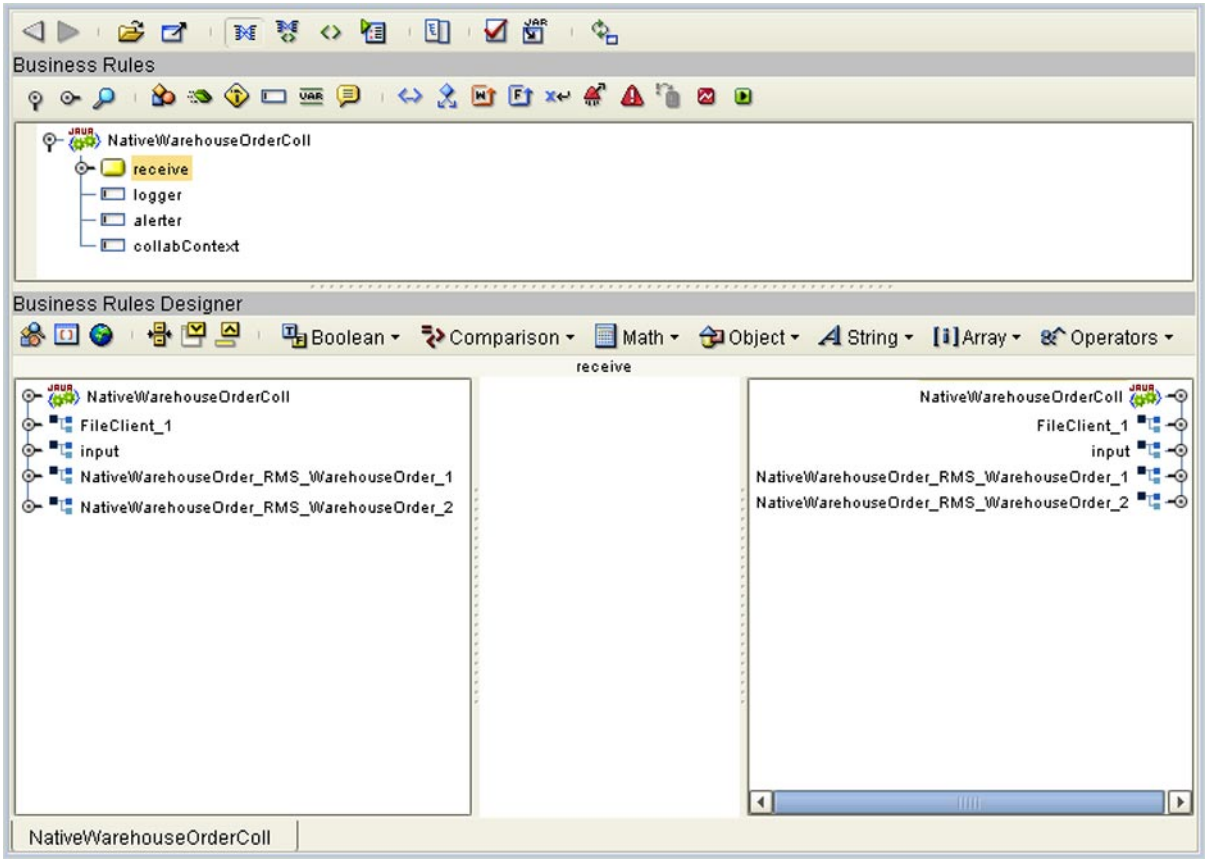
4 Click **Add** to add the OTD to the Collaboration Definition.

5 Click **Next** to proceed to the Collaboration Editor (Java).

7.3 Using the Collaboration Editor (Java)

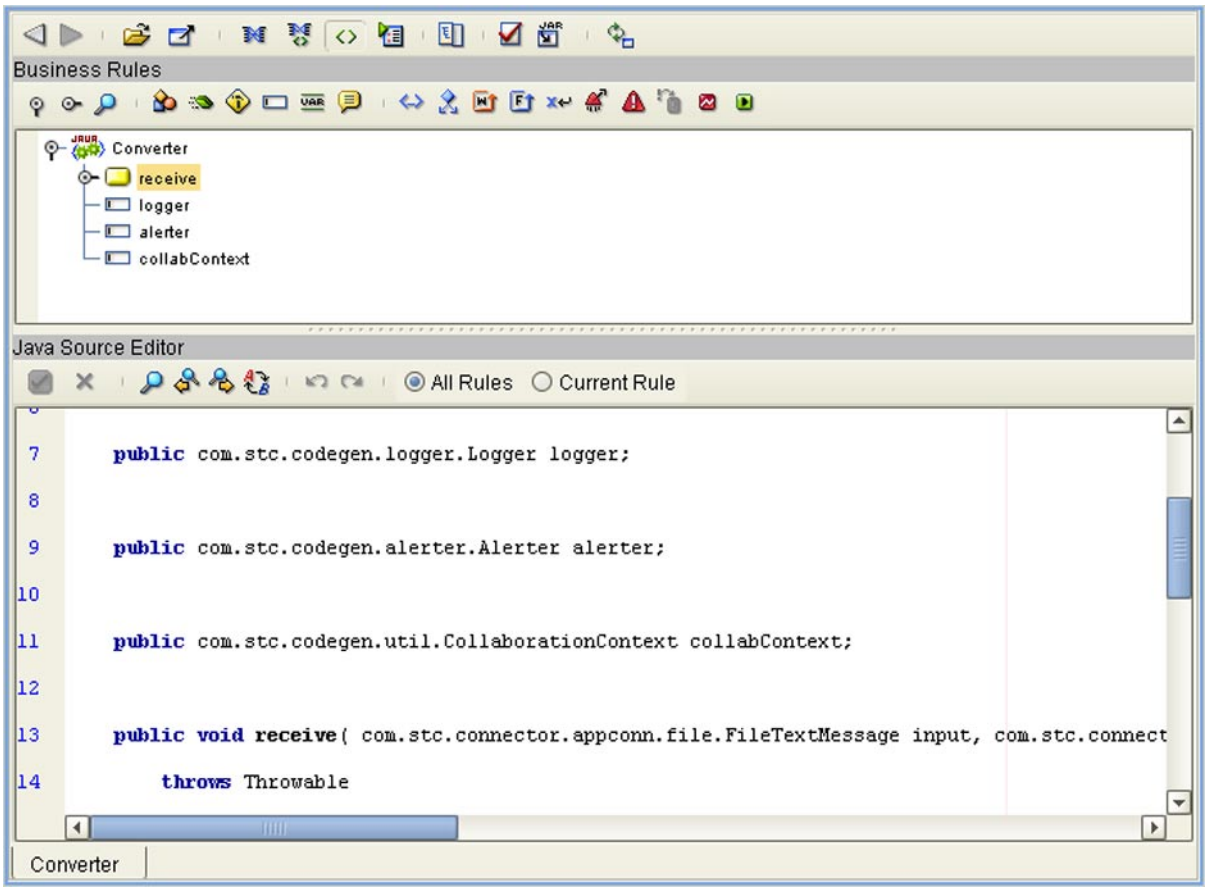
After you create a Java-based Collaboration Definition file using the Collaboration Definition Wizard (Java), the Collaboration Editor (Java) appears in the editor panel of the Enterprise Designer (see Figure 150). By default, the Business Rules Editor and Business Rules Designer panels are displayed.

Figure 150 Collaboration Editor (Java)



Selecting the *Source Code Mode* icon (see **Collaboration Editor Toolbar Icons** on page 195) substitutes the Java Source Editor for the Business Rules Designer, as shown in Figure 151.

Figure 151 Collaboration Editor (Java) - Source Code Mode



Selecting the *Advanced Mode* icon (see **Collaboration Editor Toolbar Icons** on page 195) displays both the Java Source Editor and the Business Rules Designer, in addition to the Business Rules Editor. These three parts of the Collaboration Editor (Java) are described in the following sections:

- **Business Rules Editor** on page 198.
- **Business Rules Designer** on page 203.
- **Java Source Editor** on page 201.

Important: *The Collaboration Editor (Java) does not support synchronized blocks.*

7.3.1 Collaboration Editor Toolbar Icons

Table 48 Collaboration Editor Toolbar Icons

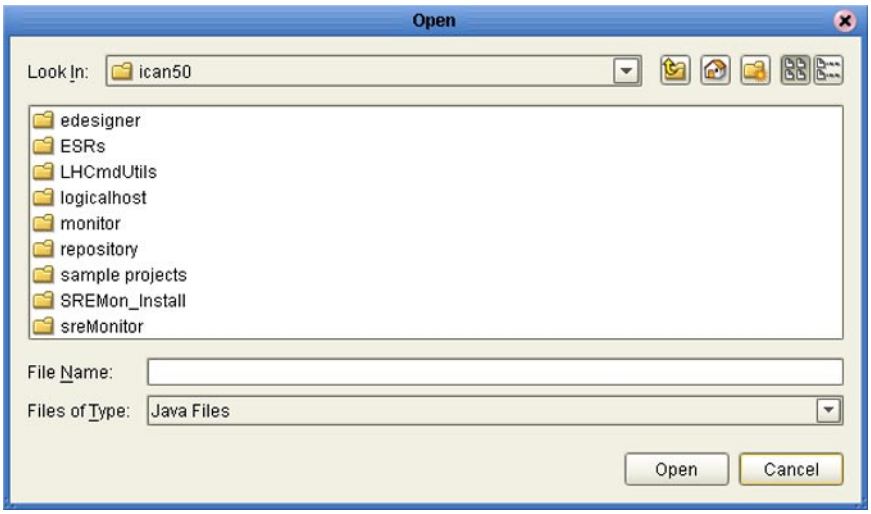
Icon	Command	Function
	Step Back	Steps back in your usage history for the current session.
	Step Forward	Steps forward in your usage history for the current session.
	Import File	Displays the **Open** dialog box, which you can use to locate and select a Collaboration Definition (Java) to import. When you import a file, any previously generated code or rules are deleted. The imported code does not get appended to the existing Collaboration Rules. 

Table 48 Collaboration Editor Toolbar Icons

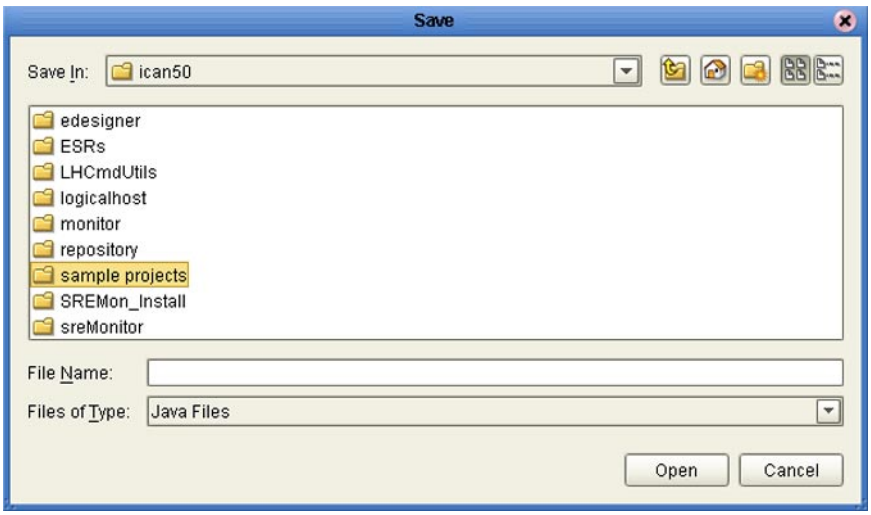
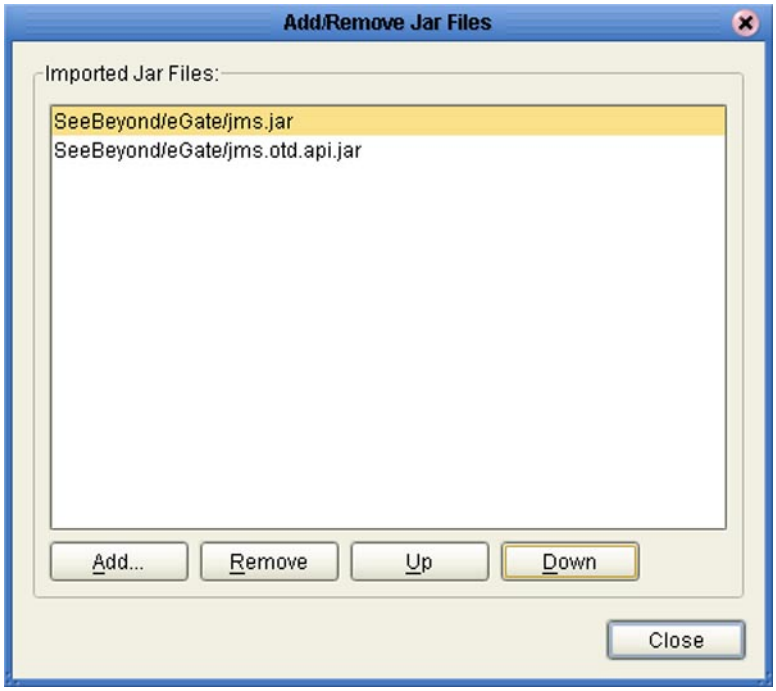
Icon	Command	Function
	Export File	Displays the **Save** dialog box, which you can use to save the selected Collaboration Definition (Java) to a file. 
	Standard Mode	Displays the Business Rules Editor and Business Rules Designer only (default setting).
	Advanced Mode	Displays the Business Rules Editor, Business Rules Designer, and Java Source Editor.
	Source Code Mode	Displays the Business Rules Editor and Java Source Editor only.
	Test Mode	Displays or hides the Java Collaboration Tester (see **Using the Collaboration Tester (Java)** on page 242).
	Business Rules on Left	Changes the editor layout to display the Business Rules panel to the left of the Business Rules Designer. Toggles with *Business Rules on Top*.
	Business Rules on Top	Changes the editor layout to display the Business Rules panel above the Business Rules Designer (default setting). Toggles with *Business Rules on Left*.
	Validate	Verifies that changes made to the Java code are valid.

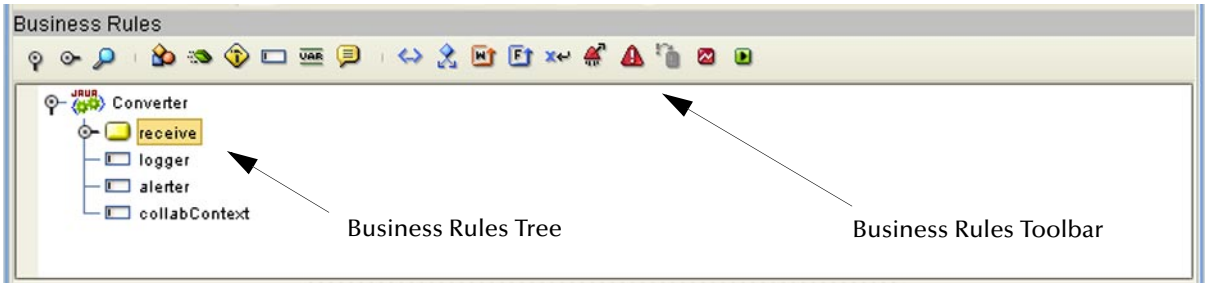
Table 48 Collaboration Editor Toolbar Icons

Icon	Command	Function
	Import JAR File	Displays the **Add/Remove Jar Files** dialog box, which you can use to import a .jar file (see **Importing Third-Party Java Classes** on page 274). 
	Refresh Collaboration	Refreshes the Collaborations from the Repository.

7.3.2 Business Rules Editor

The Business Rules Editor lists each business rule included with the Java-based Collaboration Definition. This editor has its own toolbar which you use to add business rules to the Collaboration Definition (see Figure 152). You add rules simply by dragging and dropping the rules into the Business Rules tree. Click the **Save** icon in the Enterprise Designer toolbar when you are finished.

Figure 152 Business Rules Editor



Business Rules Editor Toolbar Icons

Table 49 Business Rules Editor Toolbar Buttons

Icon	Command	Function
	Expand All Rules	Expands the tree to show all rules.
	Collapse All Rules	Collapses all rules in the tree.
	Find	Opens the **Find** dialog box, where you can enter text to search for in the Java Source Editor. 
	Class	Displays the Create Class dialog box (see **Creating Classes** on page 248).
	Method	Displays the *Create Method* dialog box, which you can use to add a new method to the Business Rules tree (see **Creating Methods** on page 250).

Table 49 Business Rules Editor Toolbar Buttons

Icon	Command	Function
	Constructor	Displays the Create Constructor dialog box (see **Creating Constructors** on page 254).
	Field	Displays the *Create Field* dialog box, which you can use to add a new field to the Business Rules tree (see **Creating Fields** on page 257).
	Local Variable	Displays the *Create a Variable* dialog box, which you can use to add a local variable to the Business Rules tree.
	Comment	Displays a dialog box in which you can enter a comment.
	Rule	Adds a new (empty) **rule** statement to the Business Rules tree. Use this command as a safeguard regarding positioning.
	If-Then	Adds an **if-then** statement to the Business Rules tree.
	While	Adds a **while** statement to the Business Rules tree, starting a specific iteration (repetition) of business rules. You can configure the condition using drag-and-drop when the **while** statement is selected.
	For	Adds a **for** statement to the Business Rules tree, starting a specific iteration (repetition) of business rules.
	Return	Adds a **return** statement to the Business Rules tree.
	Throw	Adds a **throw** statement for throwing exceptions.
	Try	Adds a **try** statement to the Business Rules tree, initiating a number of programming statements that are monitored to see whether they succeed or fail (see **Using Try-Catch** on page 278). Also enables the **Catch** icon.
	Catch	Adds a **catch** statement to an existing **try** statement (disabled in the absence of a **try** statement).
	Break	Breaks out of a business rule **while**, **for**, or **switch** loop.

Table 49 Business Rules Editor Toolbar Buttons

Icon	Command	Function
	Continue	Continues the execution of business rules in a **for** or **while** loop by starting the next iteration.

Business Rules Tree

The business rules tree consists of method and field nodes. These are the top level nodes on the tree and cannot be moved.

You add a **method** when you want to create a reusable set of instructions inside a Java-based Collaboration for a specific purpose. Methods are implemented as Java methods, and can be enhanced by means of parameters, return values, and access types (such as public, which means they are also available to other Java-based Collaborations).

You add a **field** when you want to create a *container* within a Java-based Collaboration for some specific purpose, such as storing a temporary variable. As soon as the field has been created, all other rules in the Collaboration are able to read or write from it. A field can be used with the Collaboration (access type = local) or by any other Collaboration (access type = public).

The rule, if-then, while, for, local variable, return, and try statements can be added to a method as subnodes (statements *cannot* be added to fields). The rules for placing subnodes in a method are described in Table 50.

Table 50 Rules for Placement of Subnodes

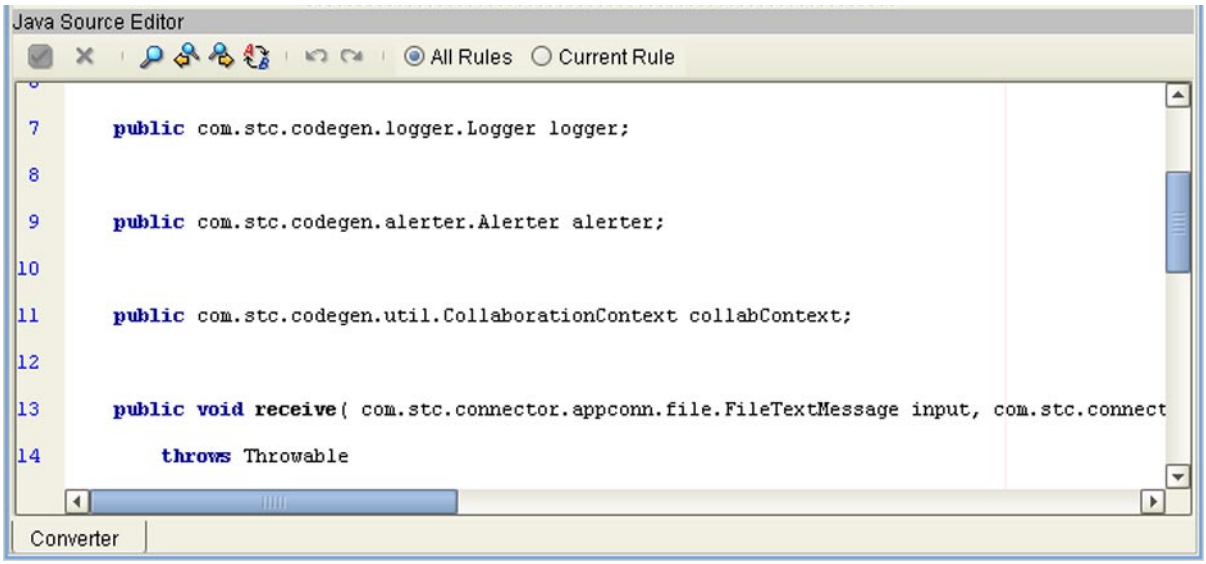
Subnode	Description
if	Place as a sibling after the last rule in the **if-then-else** block.
then	Place as a child after the last rule in the **then** code block.
else	Place as a child after the last rule in the **else** code block.
while	Place as a sibling after the last rule in the **while** node, just after the closing parenthesis.
while (code block)	The target node is represented by a code block node under the **while** node. The source rule becomes a child as the last rule in the block.
for	Place as a sibling after the last rule in the **for** node, just after the closing parenthesis.
for (code block)	The target node is represented by a code block node under the **for** node. The source rule becomes a child as the last rule in the block.
local variable	Place as a sibling after the last **local variable** block.
return	Place as a sibling after the last **return** block.
try	Place as a sibling after the last rule in the **try-catch-finally** block.
catch	Place as a child after the last rule in the **catch** code block.
finally	Place as a child after the last rule in the **finally** code block.

7.3.3 Java Source Editor

The Java Source Editor (see Figure 153) displays the actual Java code as you map the Collaboration Definition. You can display all rules, or only the rule currently selected in the Business Rules Editor, by selecting the appropriate option button in the toolbar. At your option, you can enter and edit the raw Java code for the Collaboration Definition. Click the **Commit Changes** button in the toolbar when you are finished.

Note: *You must click either the Source Code Mode or Advanced Mode toolbar icon to display the Java Source Editor.*

Figure 153 Java Source Editor



Code completion in the Java Source Editor shows standard Java library classes, classes from OTD **.jar** files, and classes from any imported third-party **.jar** files.

Java Source Editor Toolbar Icons

Table 51 Java Source Editor Toolbar Icons

Icon	Command	Function
	Commit Changes	Saves changes made in the Java Source Editor. Re-enables the Business Rules Editor and Business Rules Designer after using the Java Source Editor.
	Roll Back Changes	Cancels changes made in the Java Source Editor and returns the Java-based Collaboration Definition to the last saved state.

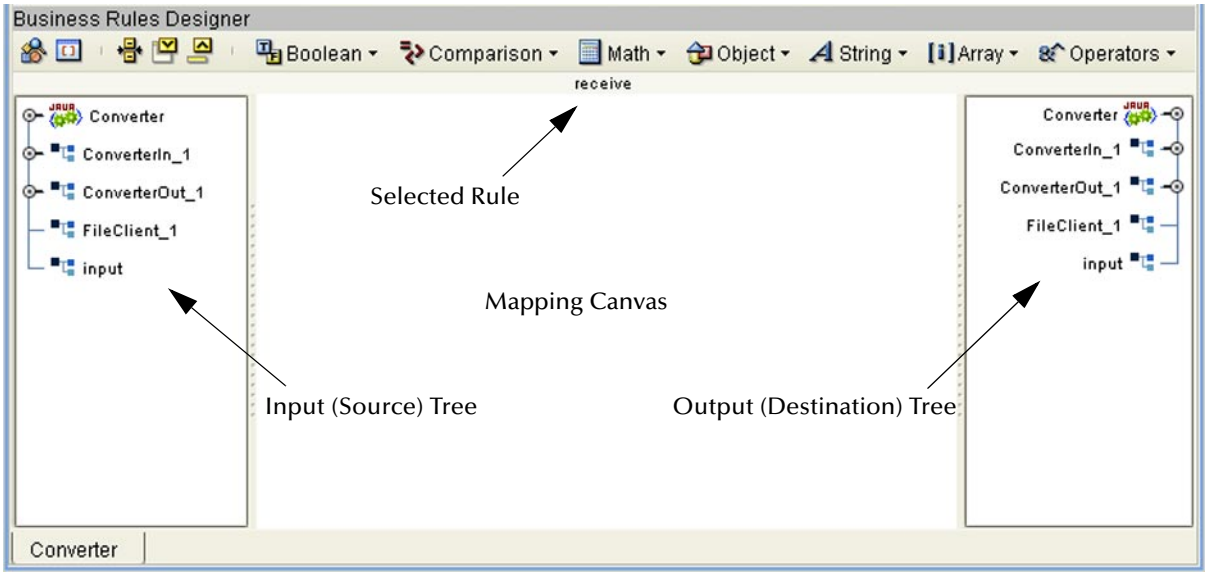
Table 51 Java Source Editor Toolbar Icons

Icon	Command	Function
	Find	Opens the **Find** dialog box, where you can enter text to search for in the Java Source Editor. 
	Find Previous	Searches backward for a previous instance of the text entered in the **Find** dialog box.
	Find Next	Searches forward for the next instance of the text entered in the **Find** dialog box.
	Replace	Opens the **Replace** dialog box, where you can enter text to search for, and to replace with, in the Java Source Editor. 
	Undo	Undoes the last operation.
	Redo	Restores the last operation, if undone.

7.3.4 Business Rules Designer

The Business Rules Designer (see Figure 154) lists each field included in the Java-based Collaboration Definition, and allows you to graphically add methods and map data paths. The Business Rules Designer has its own toolbar containing commands and menus to facilitate the creation of Collaboration Definitions (see **Business Rules Designer Toolbar Icons on page 205**).

Figure 154 Business Rules Designer



Input and output instances of all included OTDs are shown in the left and right panes, respectively. You can perform data transformation and mapping graphically on the mapping canvas, between the input and output JCD trees.

New OTDs can be added to an existing Collaboration Definition by right-clicking the Collaboration in Project Explorer to display the Collaboration Definition Properties dialog. The properties of an existing method or variable in a Collaboration can be edited by right-clicking the method or variable and modifying the properties in the resulting dialog box.

Note: *Static fields are not displayed in the JCD output tree.*

Placing Methods

Graphical representations of methods are placed on the mapping canvas in any of three ways—pick the one that best suits the occasion and your operating style:

- Right-click a node in either the source or destination tree to display the node menu and click *Select a method to call* (see **Node Menu** on page 206).
- Click a node (or a method box port) and drag the cursor onto the mapping canvas. A method dialog box appears when you release the cursor; selecting the appropriate method from the displayed list places the method box at that location (see **Using the Method Dialog Box** on page 232).
- Click a method category in the toolbar to display the corresponding drop-down menu, and click the desired option (see **Collaboration Method Menus** on page 208). The menus access a subset of all available methods.

When input and output values are directly mapped in the Business Rules Designer and the destination node is repeating or optional, the repeating/optional element in which the field is set is specified by an undefined index. This allows you to set a field in a specific element by defining the index.

Business Rules Designer Toolbar Icons

Table 52 Business Rules Designer Toolbar Icons

Icon	Command	Function
	Class Browser	Displays the Class Browser (see **Using the Class Browser** on page 227).
	New Array	Displays the *New Array* dialog box, which you can use to add an array to the Collaboration Definition (see **Creating Arrays** on page 263).
	Encoding Converter	Displays the Encoding Converter Methods dialog box (see **Defining Encoding Converter Methods** on page 282). Icon is shown only if you have enabled the *extended language options* .
	Auto Layout	Updates the layout of fields, literals, and methods that you have added to the middle column of the Business Rules Designer and vertically aligns method boxes.
	Expand All Methods	Displays the title and contents of fields, literals, and methods.
	Collapse All Methods	Displays the title of fields, literals, and methods only.
	Boolean	Displays the drop-down menu of Boolean methods.
	Comparison	Displays the drop-down menu of Comparison methods.
	Math	Displays the drop-down menu of Math methods.
	Object	Displays the drop-down menu of Object methods.
	String	Displays the drop-down menu of String methods.
	Array	Displays the drop-down menu of Array methods.
	Operator	Displays the drop-down menu of Operator methods.

Node Menu

Right-clicking a node in the source or destination tree displays the menu shown in Figure 155, which duplicates several toolbar commands for the selected node.

Figure 155 Node Menu

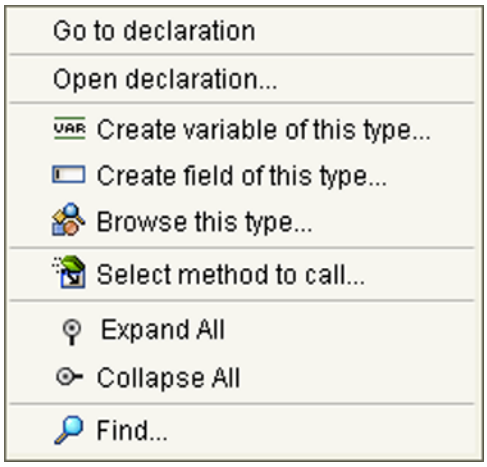


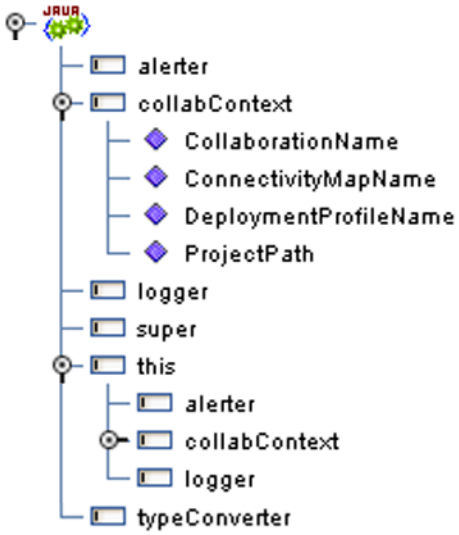
Table 53 Business Rules Designer Node Menu Options

Option	Description
Go to Declaration	Takes you to the declaration in the Business Rules editor.
Open Declaration	Opens a dialog box showing the parameters for the selected node.
Create variable of this type…	Displays the *Create Variable* dialog box (see **Creating and Modifying Variables** on page 259).
Create field of this type…	Displays the *Create Field* dialog box (see **Creating Fields** on page 257).
Browse this type…	Displays the *Class Browser* dialog box (see **Using the Class Browser** on page 227).
Select method to call…	Displays the method dialog box, listing methods appropriate to the selected node (see **Using the Method Dialog Box** on page 232).
Expand All	Expands the sub-tree under the selected node.
Collapse All	Collapses the sub-tree under the selected node.
Find	Displays the *Find* dialog box, with which you can perform a text search within the OTD tree.

Method-Access Nodes

When you create a Java-based Collaboration, several method-access nodes are added automatically in the Business Rules Designer (see Figure 156).

Figure 156 Method-Access Nodes



To call one of these methods, right-click on the node to display the node context menu and click **Select a method to call…** (see previous section).

Table 54 Method Access Nodes

Node	Subnode	Description
alerter		Allows access to **alert** methods, as described in **Generating Alerts** on page 267.
collabContext	CollaborationName	Returns the name of the selected Collaboration, as shown in the Connectivity Map.
	ConnectivityMapName	Returns the name of the Connectivity Map in which the selected Collaboration appears.
	DeploymentProfileName	Returns the name of the Deployment in which the selected Collaboration appears.
	ProjectPath	Returns the name of the Project in which the selected Collaboration appears.
logger		Allows access to **logging** methods, as described in **Creating Log Entries** on page 271.
super		Allows you to assign the class to another Java class (superclass).
this		Allows you to call other methods defined for *this* Collaboration without having to edit the Java code. Subnodes duplicate the nodes described above.

Table 54 Method Access Nodes

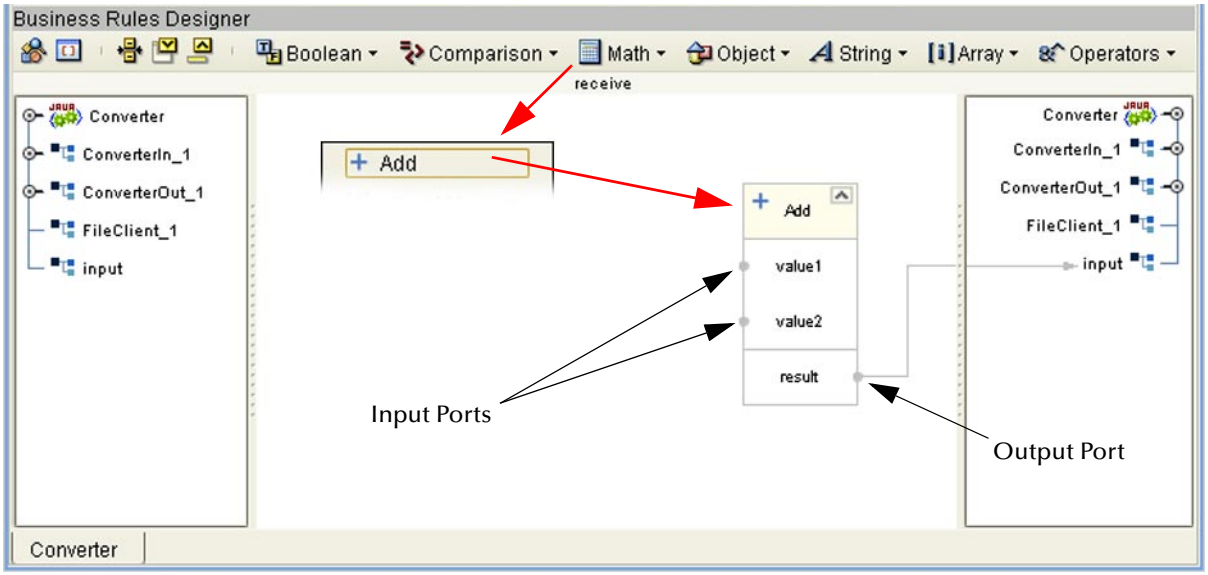
Node	Subnode	Description
typeConverter		Allows access to **datatype conversion** methods, as described in **Auto Type Conversion** on page 235.

Collaboration Method Menus

The Collaboration Method menus list commonly-used methods for use in Java-based Collaboration Definitions. Using these methods, you can create your Collaboration Definition graphically, without having to deal with Java code (see **Predefined Collaboration Methods (Java)** on page 210). Clicking a category in the toolbar displays a drop-down list of methods for that category. Clicking an individual method in the list places the corresponding, predefined method box onto the Business Rules Designer mapping canvas.

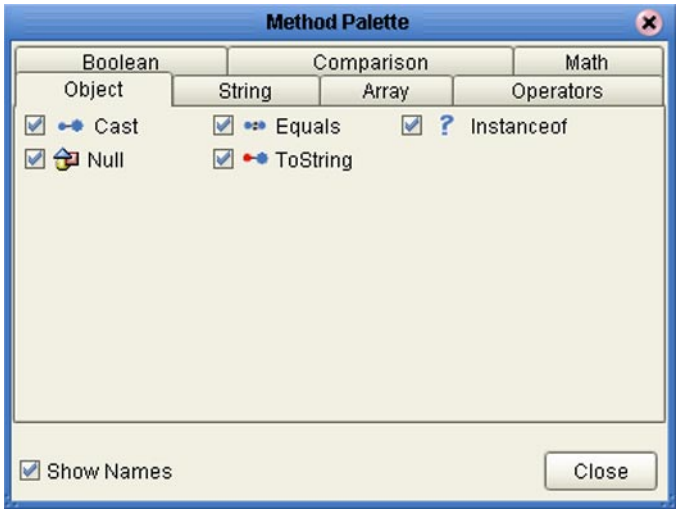
As an example, clicking **Math** in the toolbar displays a drop-down menu listing mathematical methods. Clicking **Add** in the menu places an **Add** method box on the mapping canvas. The method box contains input and output ports which you connect to the appropriate nodes in the OTD tree structures by dragging the cursor.

Figure 157 Business Rules Designer: Method Menus and Boxes



Clicking the **Settings** option in any menu displays the Method Palette dialog box shown in Figure 158, which allows you to select the methods that appear in the drop-down menus. Select a check box to add the method to the menu; clear a check box to remove the method from the menu.

Figure 158 Collaboration Method Palette (Java)



Collaboration Method Boxes

The method boxes are placed on the mapping canvas of the Business Rules Designer by clicking the method in the drop-down method menu. As shown in **Figure 157 on page 208**, the method boxes typically have input and output ports that you link to fields in the left and right panels, respectively. The method boxes are expanded by default (see Figure 159); you can collapse them (see Figure 160) by clicking the caret (^) in the upper right corner of the box. Clicking the now-inverted caret expands the box. Some boxes expand further as needed to provide additional arguments.

Figure 159 Expanded Method Box

Figure 160 Collapsed Method Box

Note: *Tooltips, derived from the associated Javadocs, appear when you hover over a field in a method box.*

7.3.5 Predefined Collaboration Methods (Java)

Boolean Methods

Figure 161 Boolean Methods Menu

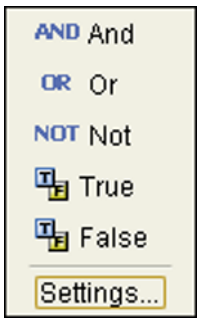


Table 55 Boolean Collaboration Methods (Java)

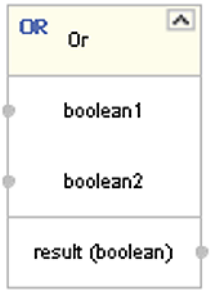
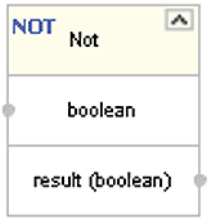
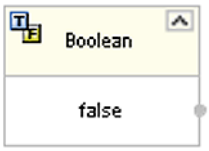
Method Box	Description/Usage
	The **And** method returns Boolean true if both *boolean1* and *boolean2* are true; otherwise, returns Boolean false.
	The **Or** method returns Boolean false if both *boolean1* and *boolean2* are false; otherwise, returns Boolean true.
	The **Not** method returns the inverse of *boolean*.

Table 55 Boolean Collaboration Methods (Java)

Method Box	Description/Usage
	The **True** method always returns Boolean true.
	The **False** method always returns Boolean false.

Comparison Methods

Figure 162 Comparison Methods Menu

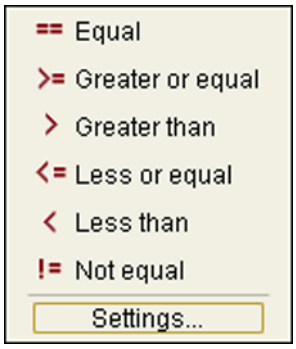


Table 56 Comparison Collaboration Methods (Java)

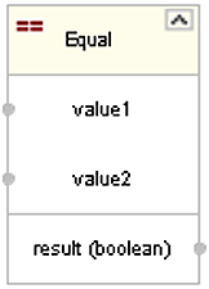
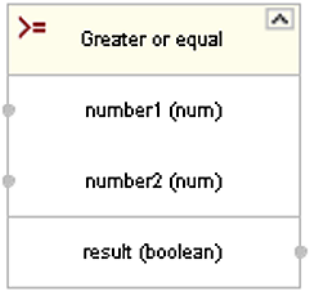
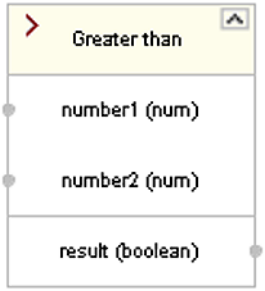
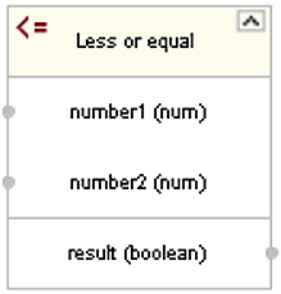
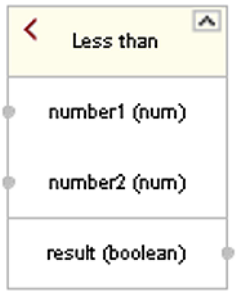
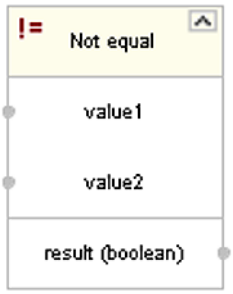
Method Box	Description/Usage
	The **Equal** method returns Boolean true if the value of *number1* is equal to the value of *number2*; otherwise, returns Boolean false.
	The **Greater or equal** method returns Boolean true if the value of *number1* is greater than or equal to the value of *number2*; otherwise, returns Boolean false.
	The **Greater than** method returns Boolean true if the value of *number1* is greater than the value of *number2*; otherwise, returns Boolean false.

Table 56 Comparison Collaboration Methods (Java)

Method Box	Description/Usage
	The **Less or equal** method returns Boolean true if the value of *number1* is less than or equal to the value of *number2*; otherwise, returns Boolean false.
	The **Less than** method returns Boolean true if the value of *number1* is less than the value of *number2*; otherwise, returns Boolean false.
	The **Not equal** method returns Boolean true if the value of *number1* is not equal to the value of *number2*; otherwise, returns Boolean false.

Math Methods

Figure 163 Math Methods Menu

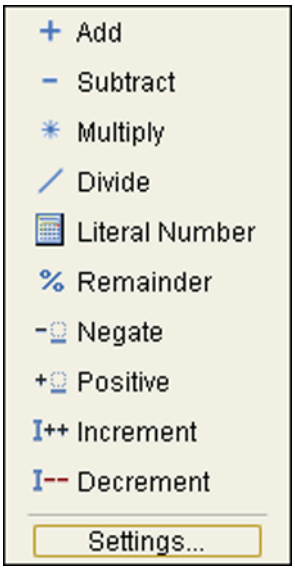


Table 57 Math Collaboration Methods (Java)

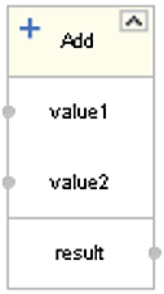
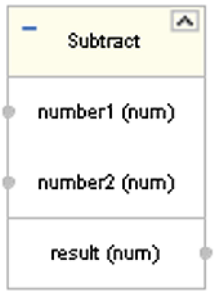
Method Box	Description/Usage
	The **Add** method returns the sum of *value1* and *value2*.
	The **Subtract** method subtracts the numerical value of *number2* from the numerical value of *number1*, returns the difference.

Table 57 Math Collaboration Methods (Java)

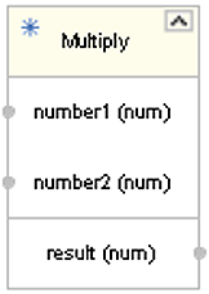
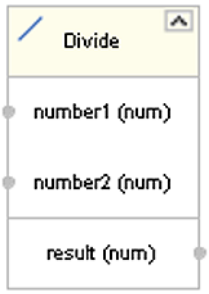
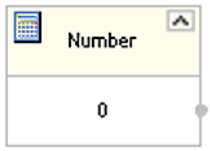
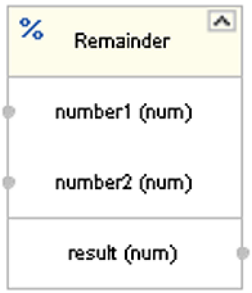
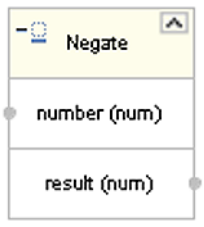
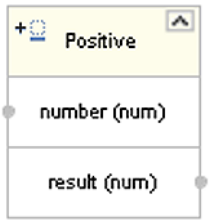
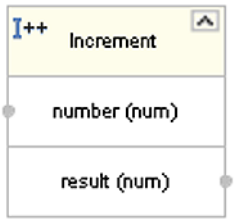
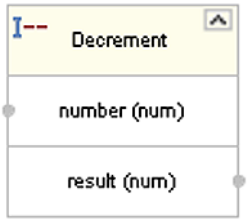
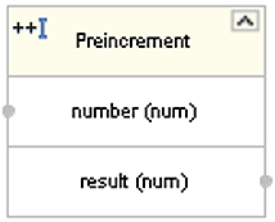
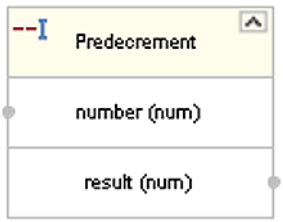
Method Box	Description/Usage
	The **Multiply** method returns the product of the numerical value of *number1* and the numerical value of *number2*.
	The **Divide** method returns the quotient of the numerical value of *number1* divided by the numerical value of *number2*.
	The literal **Number** method returns the specified numeric value.
	The **Remainder** method divides the numerical value of *number1* by the numerical value of *number2*, returns the remainder.
	The **Negate** method returns the the input number as a negative value.

Table 57 Math Collaboration Methods (Java)

Method Box	Description/Usage
	The **Positive** method returns the input number as a positive value.
	The **Increment** method increments the input number by one, following other operations.
	The **Decrement** method decrements the input number by one, following other operations.
	The **Preincrement** method increments the input number by one, preceding other operations. By default, this method does not appear in the menu—you can enable it from the method palette
	The **Predecrement** method decrements the input number by one, preceding other operations. By default, this method does not appear in the menu—you can enable it from the method palette.

Object Methods

Figure 164 Object Methods Menu

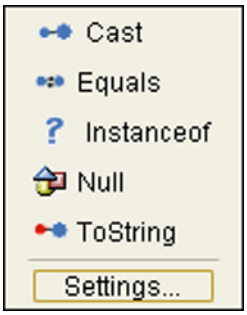


Table 58 Object Collaboration Methods (Java)

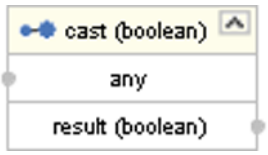
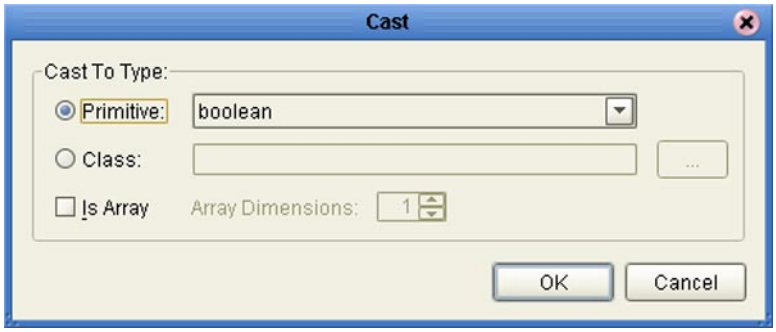
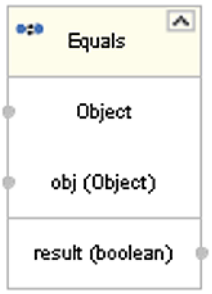
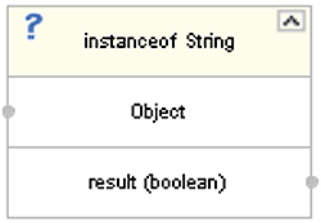
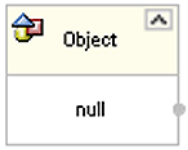
Method Box	Description/Usage
	Selecting **Cast** first displays the *Cast* dialog box, in which you specify the desired type (can be either primitive or class). Clicking **OK** places the **Cast** method box on the mapping canvas. The **Cast** method converts the given type (*any*) to the type specified in the dialog box. 
	The **Equals** method returns Boolean true if *Object* is logically equal to *obj*, otherwise, returns Boolean false.

Table 58 Object Collaboration Methods (Java)

Method Box	Description/Usage
	Selecting **Instanceof** first displays the *Class Browser* dialog box, in which you specify the desired class (see **Using the Class Browser** on page 227). Clicking **OK** places the **instanceof** method box on the mapping canvas. The **Instanceof** method returns Boolean true if *Object* is of the specified type; otherwise, returns Boolean false.
	The **Null** method always returns a null value.
	The **ToString** method converts *Object* into a string.

String Methods

Figure 165 String Methods Menu

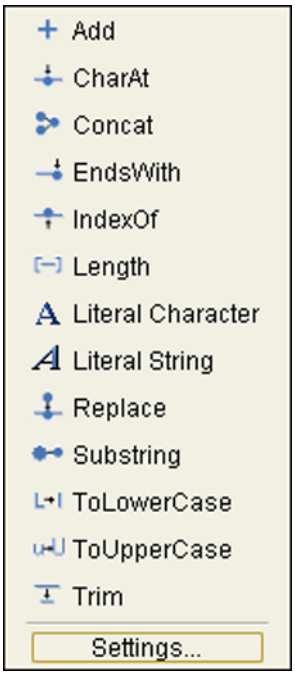


Table 59 String Collaboration Methods (Java)

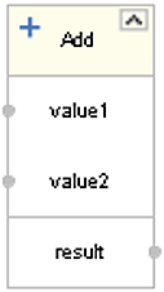
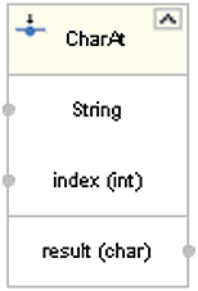
Method Box	Description/Usage
	If *value1* is a string, the **Add** method converts *value2* to a string and concatenates it to *value1* to produce the *result*.
	The **CharAt** method returns the character at the specified index, where *index* is the number of characters from the beginning of *String*.

Table 59 String Collaboration Methods (Java)

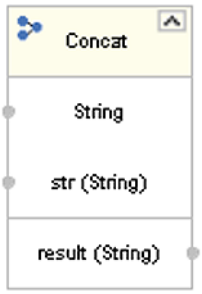
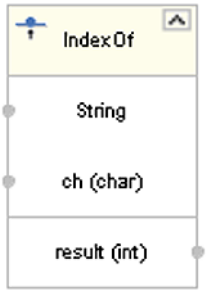
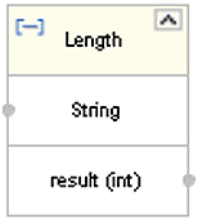
Method Box	Description/Usage
	The **Concat** method returns the string created by concatenating *str* to the end of *String*.
	The **EndsWith** method returns Boolean true if *String* ends with the string *suffix*; otherwise, returns Boolean false.
	The **IndexOf** method returns the index within *String* corresponding to the location of the specified character (*ch*), where the index represents the number of characters from the beginning of *String*.
	The **Length** method returns the length (number of characters) of *String*.

Table 59 String Collaboration Methods (Java)

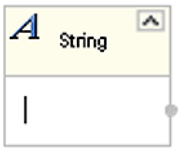
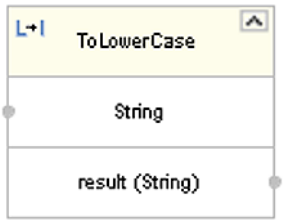
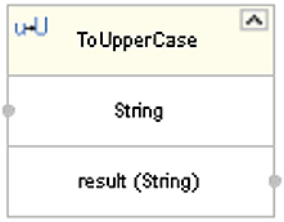
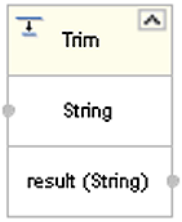
Method Box	Description/Usage
	The **Literal Char** method returns the specified character. Double-clicking in the character field enables the field for editing. Right-clicking in the string field displays a menu from which you can edit the field, or create a new variable or a new field. Note that a space character can be created by selecting **Char** and simply entering a space as a value. Although nothing will be visible in the value field of the method box, the character will appear in the Business Rules editor, as follows: 
	The **Literal String** method returns the specified string. Double-clicking in the string field enables the field for editing. Right-clicking in the string field displays a menu from which you can edit the field, or create a new variable or a new field.
	The **Replace** method returns a new string in which all occurrences of *oldChar* are replaced with *newChar*.
	The **Substring** method returns a string that is a substring of *String*, beginning from *beginIndex* (inclusive) and ending at *endIndex* (exclusive). The indices represent the number of characters from the beginning of *String*.

Table 59 String Collaboration Methods (Java)

Method Box	Description/Usage
	The **ToLowerCase** method converts all characters in *String* to lower case, using the rules of the default locale.
	The **ToUpperCase** method converts all characters in *String* to upper case, using the rules of the default locale.
	The **Trim** method trims leading and trailing whitespace from *String*.

Array Methods

Figure 166 Array Methods Menu

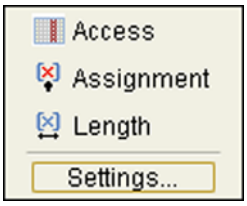


Table 60 Array Operation Collaboration Methods (Java)

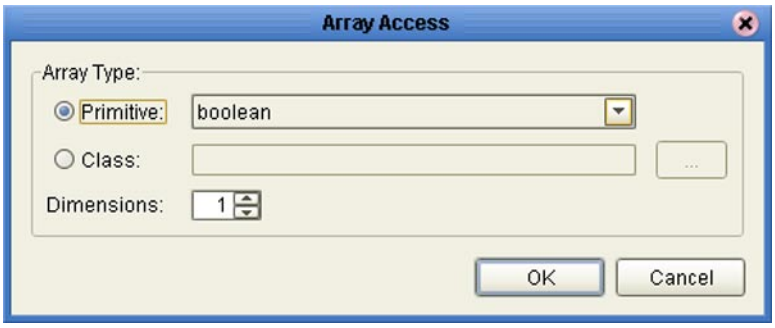
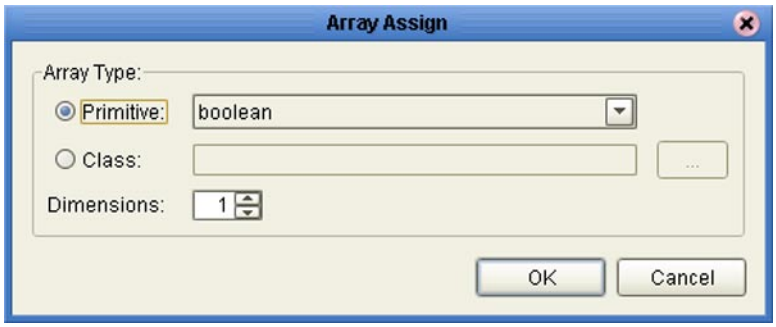
Method Box	Description/Usage
	Selecting **Access** first displays the *Array Access* dialog box, in which you specify the desired array type (can be either primitive or class) and dimensions. Clicking **OK** places the **Access** method box on the mapping canvas. The **Access** method returns the element located at the specified index within the array, where the index represents the number of elements from the beginning of the array. 
	Selecting **Assign** first displays the *Array Assign* dialog box, in which you specify the desired array type (can be either primitive or class) and dimension. Clicking **OK** places the **Assign** method box on the mapping canvas. The **Assign** method sets the element located at the specified index within the array, where the index represents the number of elements from the beginning of the array. 

Table 60 Array Operation Collaboration Methods (Java)

Method Box	Description/Usage
	The **Length** method returns the length (number of elements) of the array.

Operators

Figure 167 Operators Menu

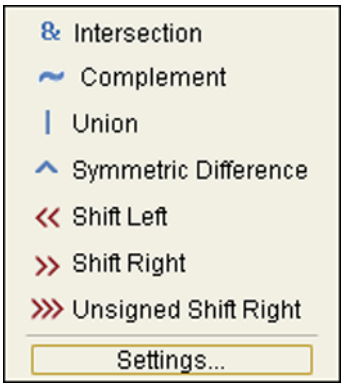


Table 61 Operators (Java)

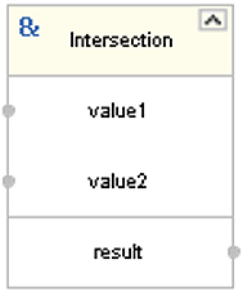
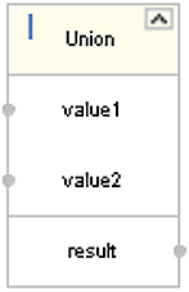
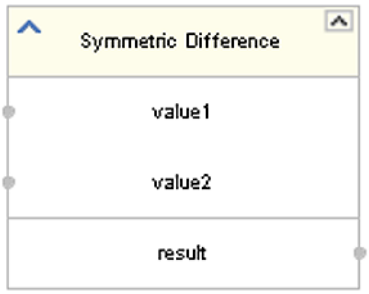
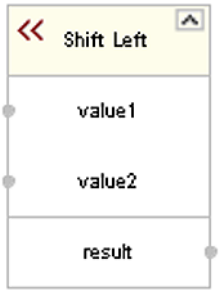
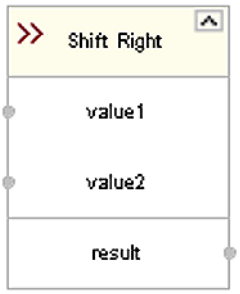
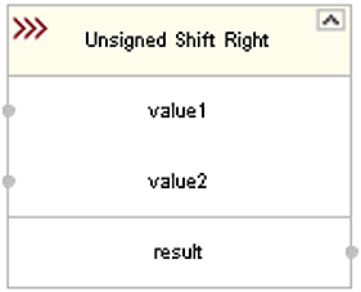
Operator Box	Description/Usage
	The **Intersection** operator computes the set intersection of bit or Boolean values. (Also listed on the Boolean menu, but not shown by default.)
	The **Complement** operator computes the complement of a bit value by inverting each bit of the input number.
	The **Union** operator computes the union of the input values. (Also listed on the Boolean menu, but not shown by default.)

Table 61 Operators (Java)

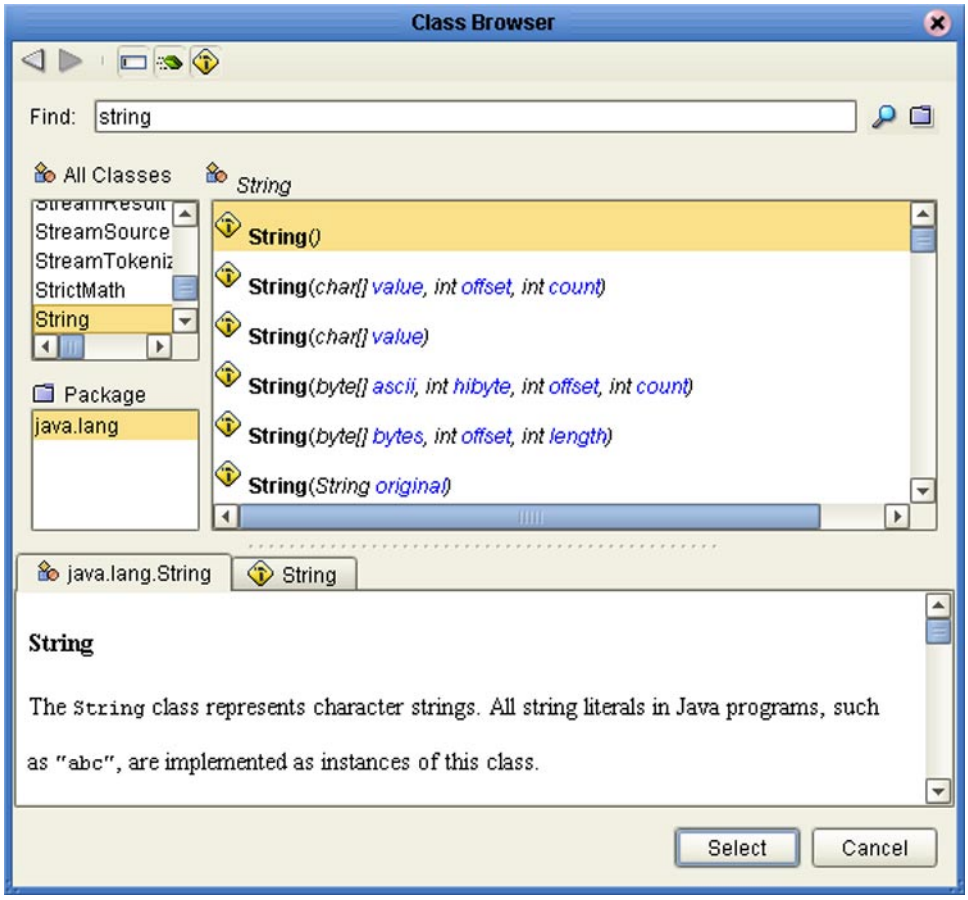
Operator Box	Description/Usage
	The **Symmetric Difference** operator computes the set symmetric difference of bit or Boolean values. (Also listed on the Boolean menu, but not shown by default.)
	The **Shift Left** operator shifts bits to the left; equivalent to multiplication by power of two.
	The **Shift Right** operator shifts bits to the right; new left bits are filled with the value of the high-order bit so the sign is retained.
	The **Unsigned Shift Right** operator shifts bits to the right, based on the value of the right operand; new left bits are filled with zeros. For non-negative integers, an unsigned right shift is equivalent to dividing the left operator by the number two raised to the power of the right operator.

7.4 Using the Class Browser

Clicking the Class Browser icon in the Business Rules Designer toolbar invokes the Class Browser, which you can use to search for Java classes and methods. The browser displays all available classes, including any third-party classes that have been uploaded.

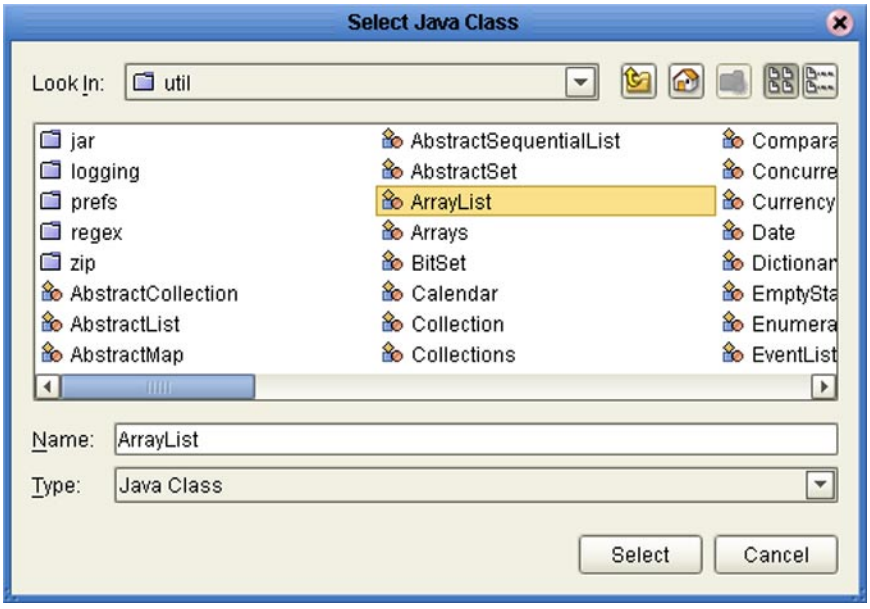
Figure 168 Class Browser Icon

Figure 169 Class/Method Browser



7.4.1 Class Browser Toolbar Icons

Table 62 Method Browser Toolbar Icons

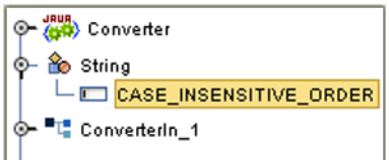
Icon	Command	Function
	Step Back	Steps back in your usage history for the current session.
	Step Forward	Steps forward in your usage history for the current session.
	Show Fields	Displays or hides all fields in the selected class. By default, all fields are displayed; you must click the icon to remove them from the list.
	Show Methods	Displays or hides all methods in the selected class. By default, all methods are displayed; you must click the icon to remove them from the list.
	Show Constructors	Displays or hides all constructors in the selected class. By default, all constructors are displayed; you must click the icon to remove them from the list.
	Search	Executes search on an entry in **Find** text box.
	Browse Java Package Hierarchy	Displays the **Select Java Class** dialog box, where you can locate classes contained in another installed package. Clicking **Select** displays the contents of the package in the Class Browser. 

You can locate a class by scrolling through the class list or by typing the first few letters of a class name into the text box and clicking the **Search** icon. Scrolling through the method list displays all methods, constructors, and fields defined for that class. If the desired class is not listed, you can search for it in other installed packages by using the **Browse Java Package Hierarchy** icon.

The action resulting from clicking **Select** depends upon how you are using the browser.

- If you are adding a class or method instance to the Collaboration, it adds a constructor or method box to the Business Rules Designer mapping canvas (see **To add a class instance using the Class Browser** on page 230).
- If you are adding a field to the Collaboration, it adds the class and field to the source OTD, as shown in Figure 170.

Figure 170 Class and Field



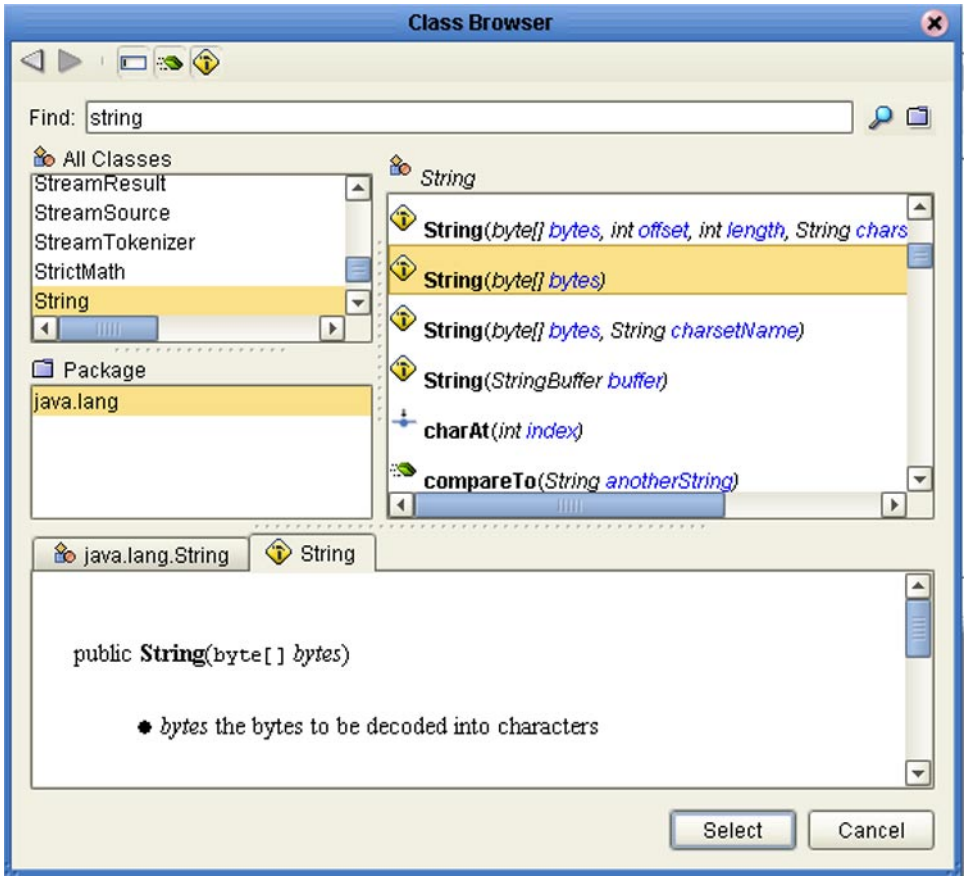
- If you are creating a variable, it adds the variable to the tree in the Business Rules editor (see **Creating and Modifying Variables** on page 259).

To add a class instance using the Class Browser

1 Click the **Class Browser** icon in the Business Rules Designer toolbar to display the Class Browser.

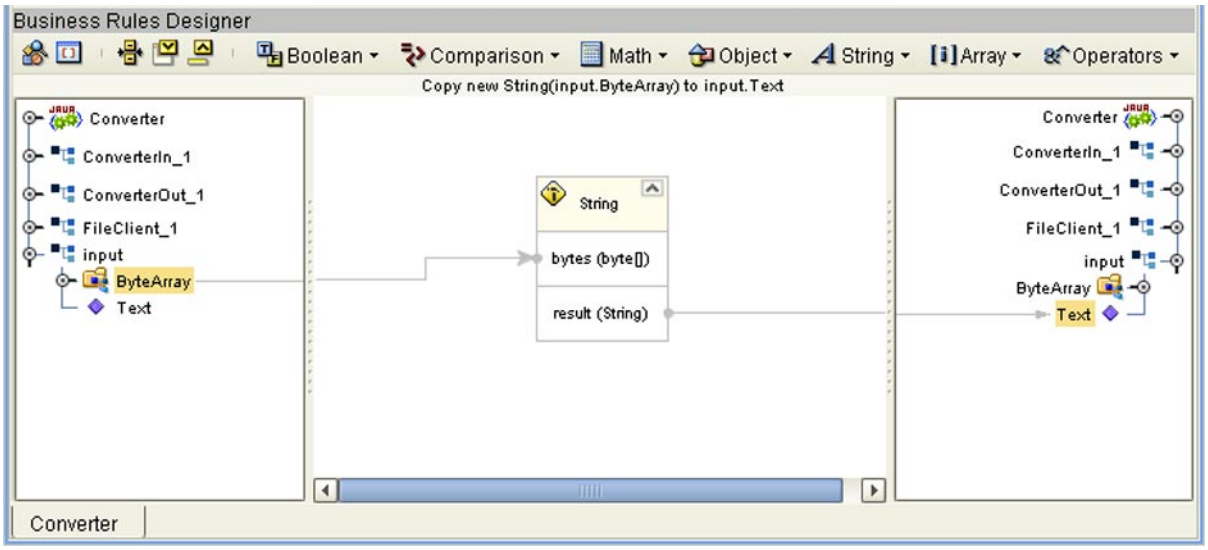
2 In the Class Browser, search for and select the desired class and constructor (see Figure 171).

Figure 171 Class Browser - Select Constructor



3 Click **Select** to place the constructor box on the Business Rules Designer mapping canvas as shown in Figure 172.

Figure 172 Constructor Box on Mapping Canvas



4 Link the constructor box ports to the appropriate OTD nodes, as shown.

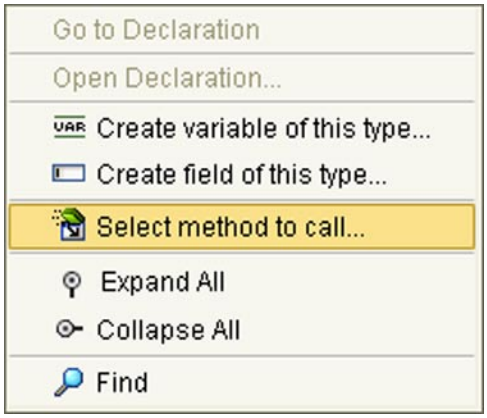
5 **Save** to the Repository.

7.5 Using the Method Dialog Box

The method dialog box provides an alternative to the Class Browser for selecting methods and constructors, and also provides for creating literals. To display the method dialog box, select a node in either the left or right panel of the Business Rules Designer (or a method box port) and:

- Right-click on the node to display the context menu and click **Select a method to call** (see Figure 173), or
- Left-click on the node and drag the cursor to the mapping panel.

Figure 173 Node Context Menu



Either action will display the method list for that specific node type, such as that shown in Figure 174 (using the second option allows you to select the position of the resulting method box, which can save time). A description of the method appears in the lower panel of the method dialog box.

Figure 174 Method Dialog Box (for type *TextMessage*)

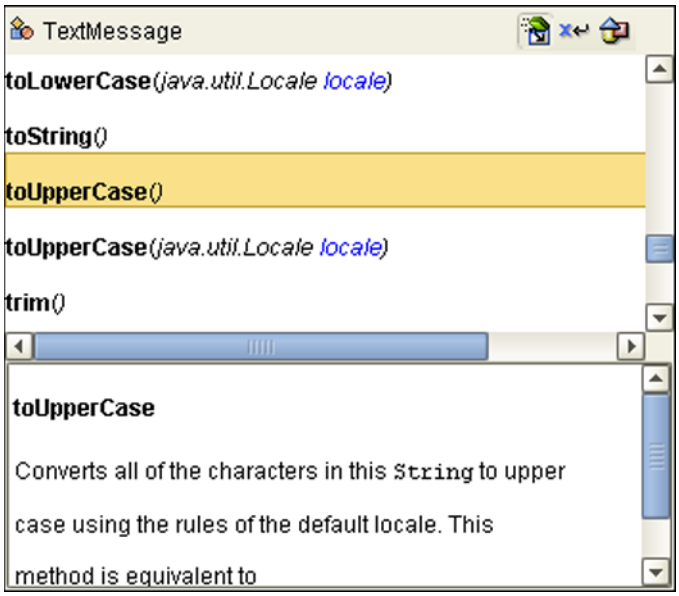


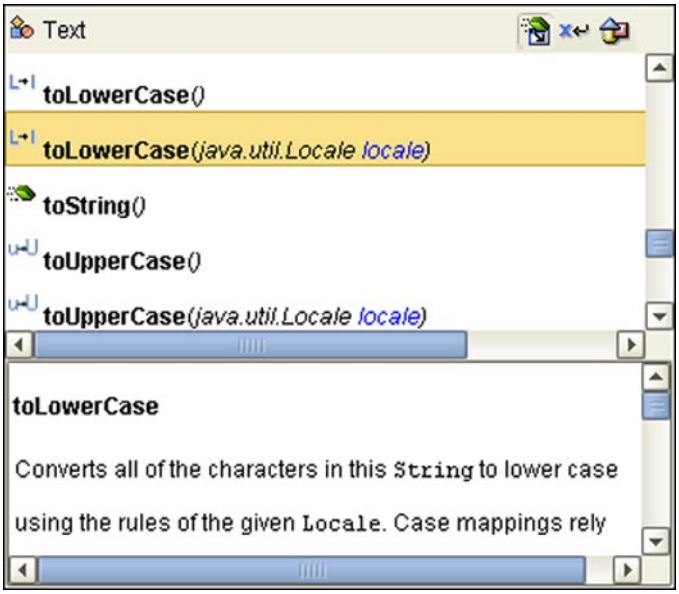
Table 63 Method Dialog Box Toolbar Icons

Icon	Command	Function
	Show Methods	Displays a list of methods **having** the same type as the selected node. This is the default display when the dialog box is launched from the *left* panel of the Business Rules Designer or a method box *output* port.
	Return	Displays a list of methods **returning** the same type as the selected node. This is the default display when the dialog box is launched from the *right* panel of the Business Rules Designer or a method box *input* port.
 **OR** 		Creates a literal of the same type as the as the selected node (object or string), placing the literal box on the mapping canvas for linking. See **Creating and Modifying Literals** on page 265.

To add a method instance using the method dialog box

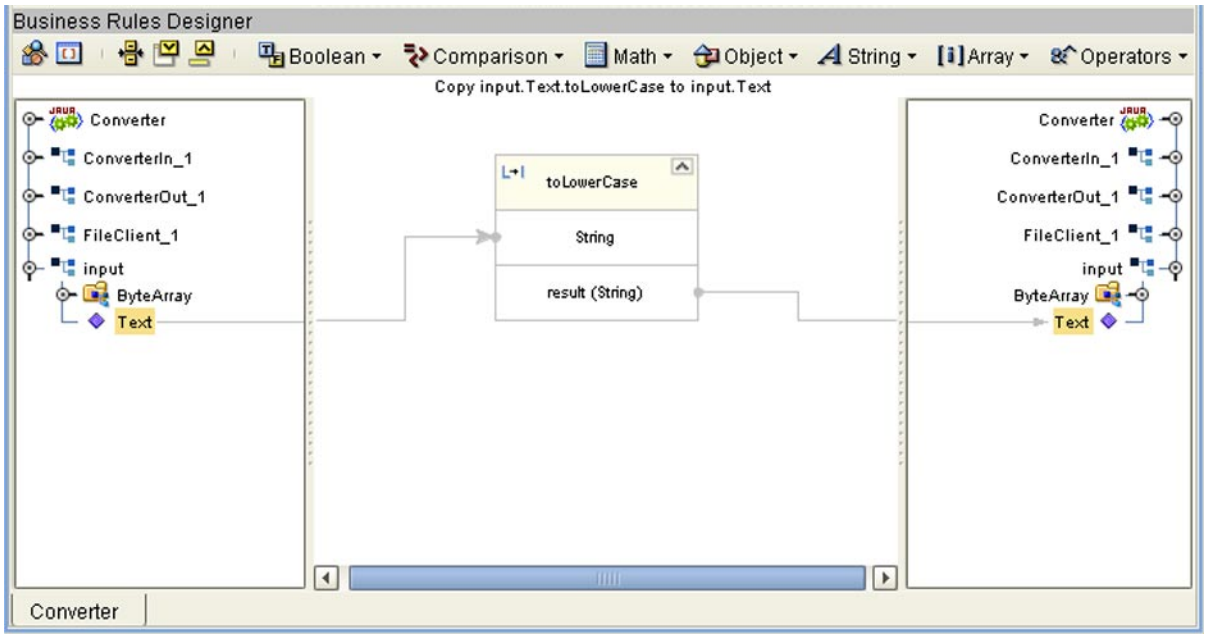
1 To display a list of methods of the same type as a node (see Figure 175), either right-click on the node (and release) or left-click on the node and drag to the mapping panel.

Figure 175 Method List for *Text*



2 Scroll to the desired method and double-click on the method to place the method box in the Business Rules Designer as shown in Figure 176, then link the method ports and **Save** to the Repository.

Figure 176 Method Box in Mapping Panel



7.6 Auto Type Conversion

When you map between OTD nodes having different data types, if a conversion method exists for that pair of data types, a dialog box is automatically displayed in which you can specify certain properties associated with the conversion. The primary content of the dialog box is specific to the conversion, examples of which are described in following sections. Two properties that are common to all dialog boxes deal with exception handling and whether or not the dialog box is displayed for each conversion.

Exception handling

Select the method for handling exceptions, either *Use Default Value* or *Throw Exception*, from the drop-down menu (see Figure 177). In general,

- If you select **Use Default Value**, the output will be based on the unquoted string in the *Default Value* box if the desired conversion cannot be performed. See the individual conversion type descriptions for specific details.
- If you select **Throw Exception**, then the code should use a *try/catch* block to catch the exception should it occur at run-time.

Figure 177 Exception Handling Options

Automatic conversion

If you want your specification to be used for all subsequent conversions between the same data types, select the check box beside **Please convert automatically next time…** (it is selected by default—see Figure 178). If this option is enabled, all subsequent conversions between those same data types will use the same settings, and the dialog box will not be displayed.

Note: *The dialog box will appear again the next time you open the Collaboration Definition Editor—you can reset the option at that time.*

Figure 178 Automatic Conversion Option

If you want to specify each conversion individually, clear the check box. The dialog box will then be displayed for each conversion.

Data types

Not all data types can be converted into other specific data types— conversions for which type conversion methods exist are listed in Table 64. You can also access these methods from the **typeConverter** node (see **Method-Access Nodes** on page 207). Attempting to map between data types for which no type conversion method exists displays an error notice.

Table 64 Supported Datatype Conversions

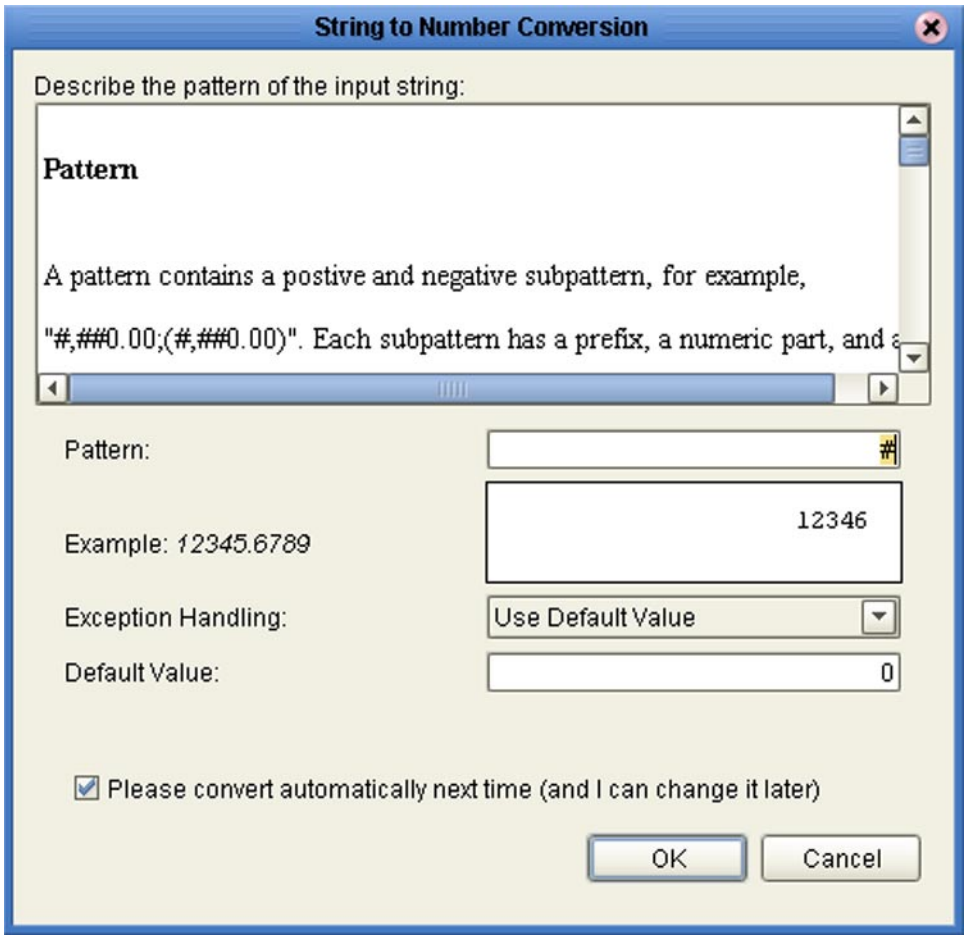
String to Byte Array	Byte Array to String
String to Date	Date to String
String to Double	Double to String
String to Float	Float to String
String to Int	Int to String
String to Long	Long to String
String to Short	Short to String
String to SQL Date	String to Timestamp

7.6.1 Character Conversion

String to Number Conversion

Mapping between a *Text* node and an *int* node automatically displays the dialog box shown in Figure 179, where you can specify the pattern of the input string .

Figure 179 Type Conversion - String to Number



String Pattern

You specify the pattern by entering characters in the **Pattern** text box, and an example illustrating the resulting output appears in the field below. A comprehensive explanation of how this works is provided in the scrollable text box above. For your convenience, the following table lists special characters that are frequently used.

Table 65 Special Characters

Symbol	Location	Description or Use	Use Quotes for Literal?
0	Number	Digit.	Yes
1-9	Number	'1' - '9' indicate rounding.	Yes
@	Number	Significant digit.	Yes
#	Number	Digit (zero shows as absent).	Yes
.	Number	Decimal separator.	Yes
-	Number	Minus sign.	Yes
,	Number	Grouping separator.	Yes
E	Number	Separates mantissa and exponent in scientific notation.	No
+	Exponent	Prefix positive exponents with localized plus sign.	Yes
;	Subpattern boundary	Separates positive and negative subpatterns.	Yes
%	Prefix or suffix	Multiply by 100 and show as percentage.	Yes
/u2030	Prefix or suffix	Multiply by 1000 and show as per mile.	Yes
'	Prefix or suffix	Used to quote special characters in a prefix or suffix to appear as literal.	See below
*	Prefix or suffix boundary	Pad escape, precedes pad character.	Yes

To be entered as literals, special characters generally must be entered within quotation marks, but there are exceptions to this rule, as noted in the table. To have a single quote (') appear as a literal (as an apostrophe, for example), precede it with another single quote.

Exception Handling

This parameter allows you to specify that when the desired conversion cannot be performed, either:

- An exception should be thrown, or
- The output should be set to the value of the default string specified as the *Default Value* parameter.

Default Value

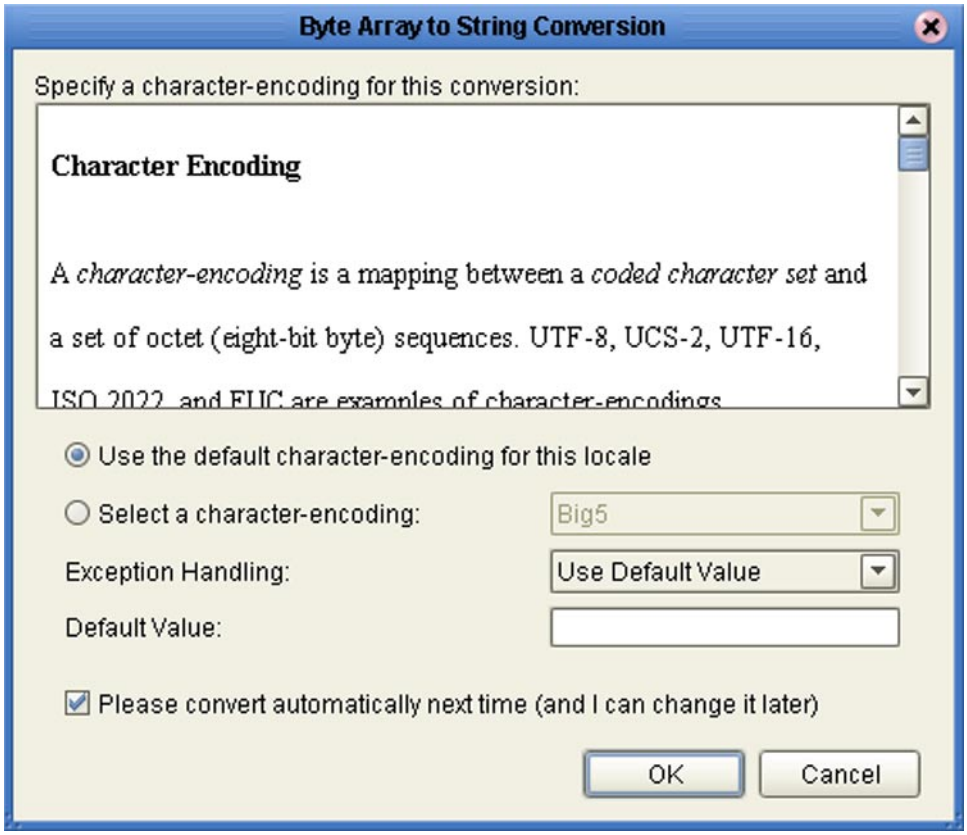
If you have selected *Default Value* for the exception handling, specify an unquoted character string. The output string will be set to this string value.

7.6.2 Character Encoding

Mapping between a *ByteArray* node and a *Text* node automatically displays a dialog box such as that shown in Figure 180, where you can specify the character encoding.

Note: *This encoding is independent of the encoding specified for the OTD.*

Figure 180 Type Conversion - Character Encoding



Byte-to-String Conversion

Conversion Options

When converting a byte array to a Java string, the resulting string is defined by the character encoding. Please select from the following options:

- **Use the default character-encoding for this locale**

  Selected by default, this option makes use of the default locale encoding, as set in the Java Virtual Machine containing the Logical host. You should select this option if you are dealing with a single encoding for all your data, and your Java Virtual Machine encoding is known to be correct.

- **Select a character-encoding**

  Select this option to manually specify the encoding of your (known) byte[] data from the drop-down list of encodings supported by this version of Java.

Exception Handling

Java strings are encoded within Java in UTF-16 unicode. When converting from a byte[] to a Java string, if the encoding you specify does not match the actual encoding of the byte[] data (for example, if you are expecting UTF-8 data and the byte[] you receive from an external source actually contains double-byte data in some other encoding—such as Shift_JIS), then the conversion will not produce the correct Java string.

This parameter allows you to specify that when any invalid byte values are encountered in the conversion, either:

- An exception should be thrown, or
- The entire byte[] should be set to the byte value of the default string specified as the *Default Value* parameter.

Default Value

If you have selected *Default Value* for the exception handling, specify an unquoted character string. The output string will be set to this string value.

String-to-Byte Conversion

Conversion Options

When converting a Java string to a byte array, the byte value for each character is defined by the character encoding used. Please select from the following options:

- **Use the default character-encoding for this locale**

  Selected by default, this option makes use of the default locale encoding, as set in the Java Virtual Machine containing the Logical host. You should select this option if you are dealing with a single encoding for all your data, and your Java Virtual Machine encoding is known to be correct.

- **Select a character-encoding**

  Select this option to manually specify the encoding expected by your application from the drop-down list of encodings supported by this version of Java.

Exception Handling

Java strings are encoded within Java in UTF-16 unicode. When converting from a Java string to an encoding other than UTF-16, there may be some characters in the string that are not included in the target character encoding.

This parameter allows you to specify that when any such character is encountered in the conversion, either:

- An exception should be thrown, or
- The entire byte[] should be set to the byte value of the default string specified as the *Default Value* parameter.

Default Value

If you have selected *Default Value* for the exception handling, specify an unquoted character string. The output byte array will be set by converting this string to a byte[], using the encoding you selected above.

Note: *For an understanding of character encoding in general, please refer to* **http://java.sun.com/j2se/1.4.2/docs/guide/intl/index.html**. *Also,* ***CJVK Information*** *by Ken Lunde (O'Reilly) provides a detailed explaination of character encoding.*

7.7 Using the Collaboration Tester (Java)

The Collaboration Tester (Java) supports testing of both marshalable and unmarshalable OTDs and, combined with the Java Source Editor, allows line-by-line debugging of the Java code.

Note: *See also* **Using the Java Debugger** *on page 298 for a description of the Java Debugger, which works with deployed Collaborations.*

Selecting the **Test Mode** icon in the main Collaboration Editor toolbar (see Figure 181) displays the Collaboration Tester panel, which appears at the left of the Business Rules editor (see Figure 182).

Figure 181 Test Mode Icon

Note: *The function of this icon toggles between **display** and **hide**, depending upon the current state of the tester.*

Figure 182 Collaboration Tester (Java)

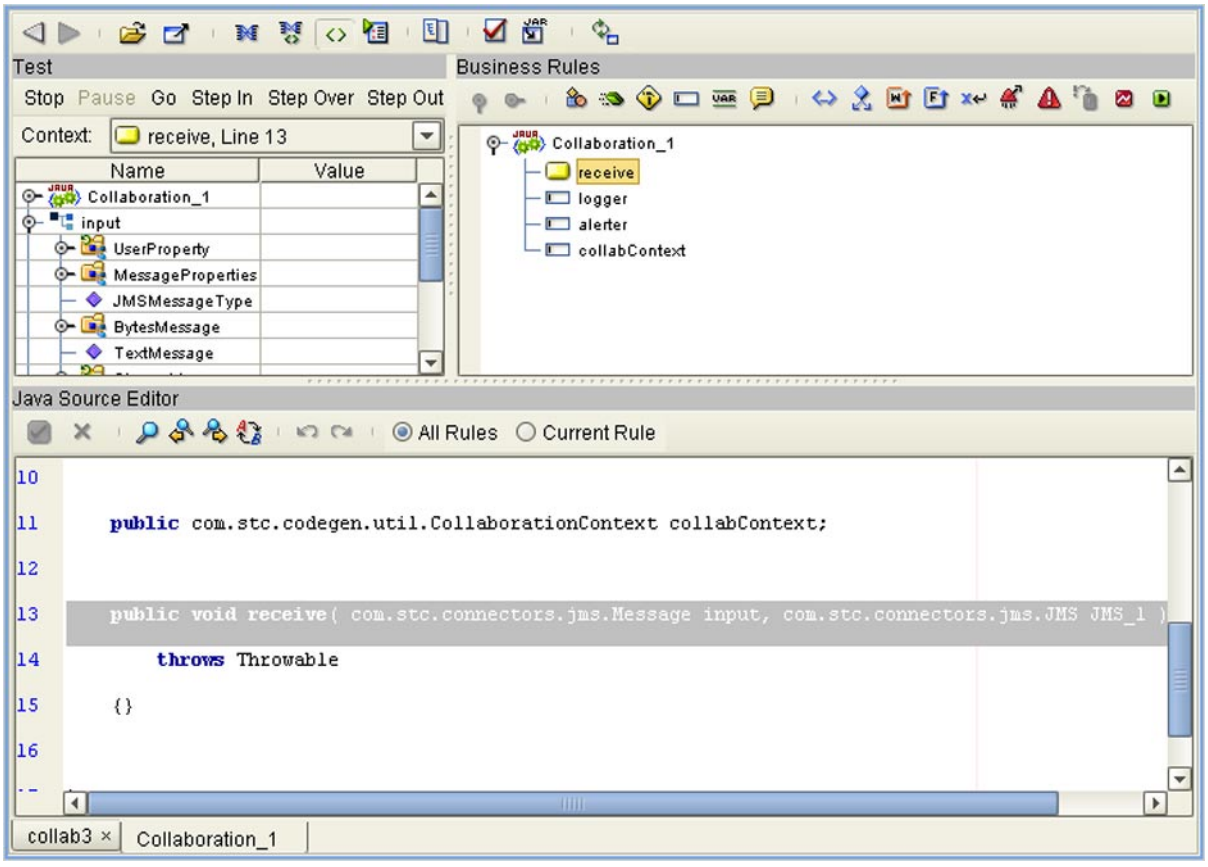


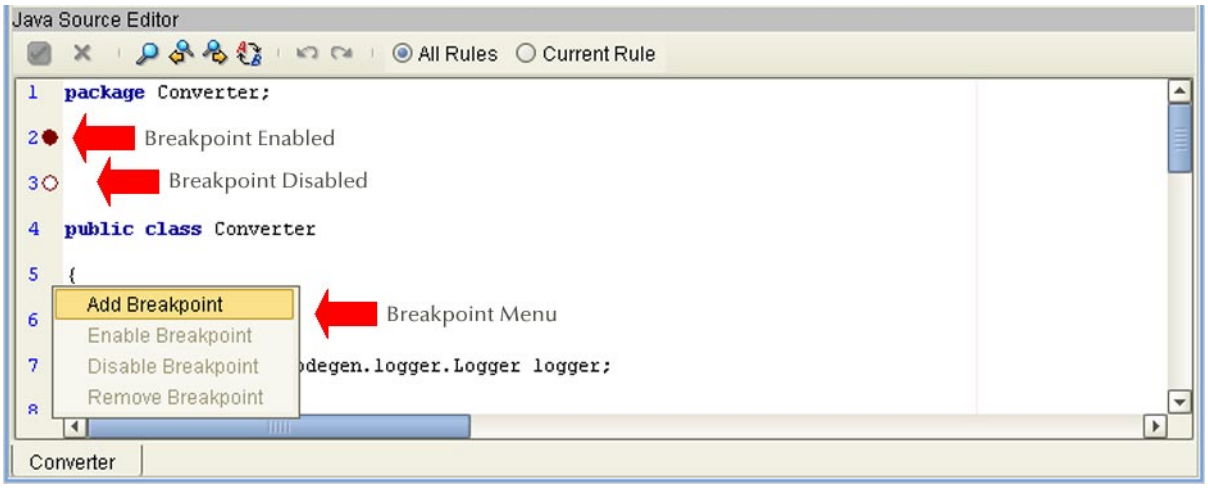
Table 66 Collaboration Tester (Java) Commands

Command	Description
Start / Stop	Toggles to start or stop test procedure. After stopping, clicking **Start** refreshes the tree.
Pause	Pauses test procedure on current line.
Go	Proceed to next step.
Step In	Steps into Java code by one hierarchical level.
Step Over	Steps to next line of code, at same hierarchical level.
Step Out	Steps out of Java code by one hierarchical level.

7.7.1 Breakpoints

You can set breakpoints in the Java code where you want execution to stop during testing. A single click next to a line number in the Java Source Editor adds and enables a breakpoint, placing a red dot marker in the left column. Clicking on a red dot disables the breakpoint, changing the dot to white. Clicking on a white dot removes the breakpoint. The markers are illustrated in Figure 183.

Figure 183 Breakpoints in Java Code



Alternatively, right-clicking next to a line number displays a breakpoint menu from which you can select the desired action (see Figure 184). In the absence of an existing breakpoint, the menu appears as shown in Figure 183. Right-clicking on an existing breakpoint enables the other options, as appropriate.

Figure 184 Breakpoint Menu

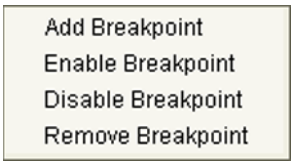


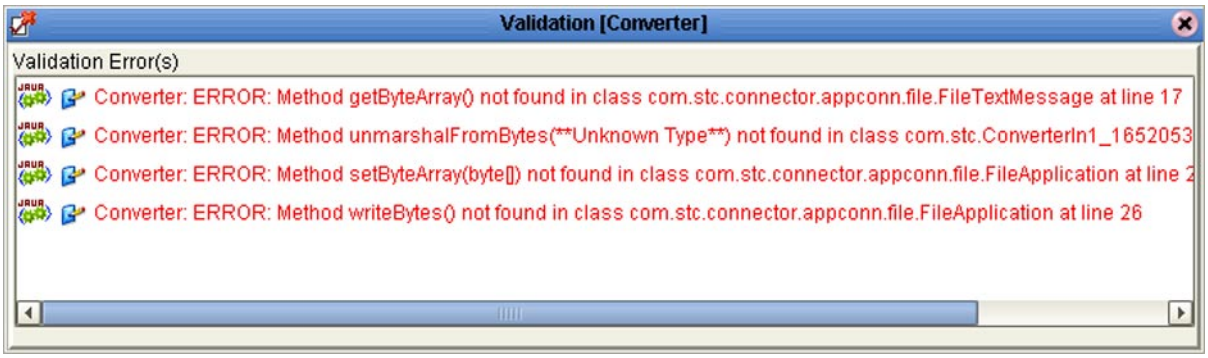
Table 67 Breakpoint Menu Options

Option	Action
Add Breakpoint	Adds and enables a breakpoint at the selected line.
Enable Breakpoint	Enables a disabled breakpoint.
Disable Breakpoint	Disables an enabled breakpoint.
Remove Breakpoint	Removes the breakpoint.

7.7.2 Code Validation

The first step in testing is to validate the Java code. If errors are found, a *Validate* panel is displayed listing all errors (seeFigure 185). The test will not proceed until these errors are resolved.

Figure 185 Validation Panel



To test an OTD using the Collaboration Tester (Java)

1 Open a Collaboration to display the Collaboration Editor.

2 Click the **Source Code Mode** icon to display the Source Code Editor.

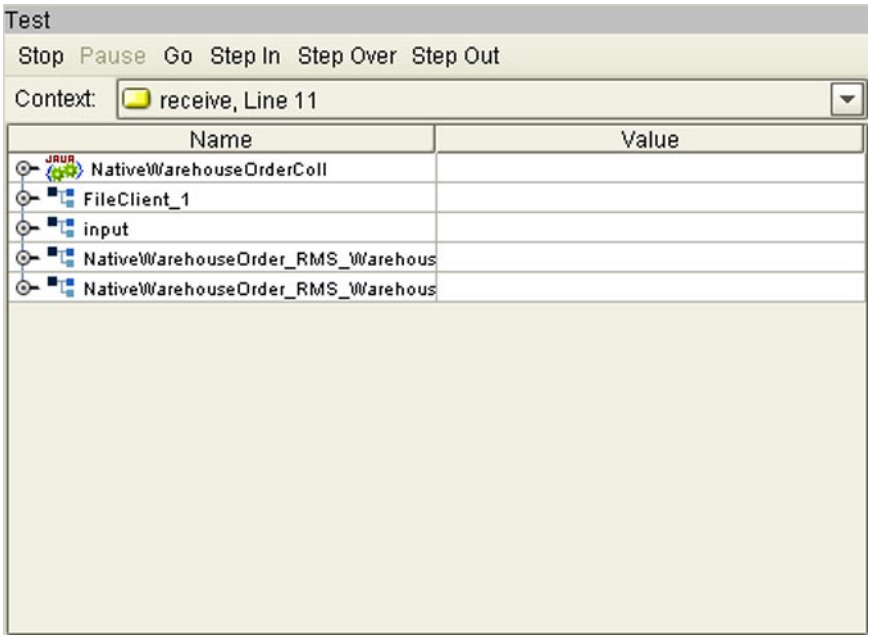
Figure 186 Source Code Mode Icon

3 Click the **Test Mode** icon to display the Collaboration Tester.

Figure 187 Test Mode Icon

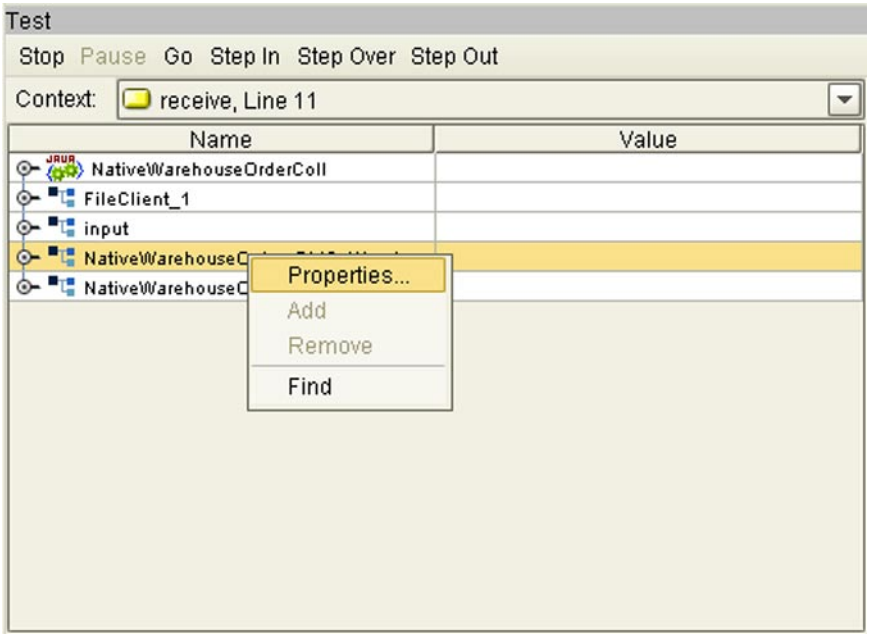
4 In the *Test* panel, click **Start** to initialize the Tester. The Collaboration tree will be displayed (see Figure 188), and the Tester will advance to the first rule (highlighted in the *Source Code* panel).

Figure 188 Initial Collaboration Tree



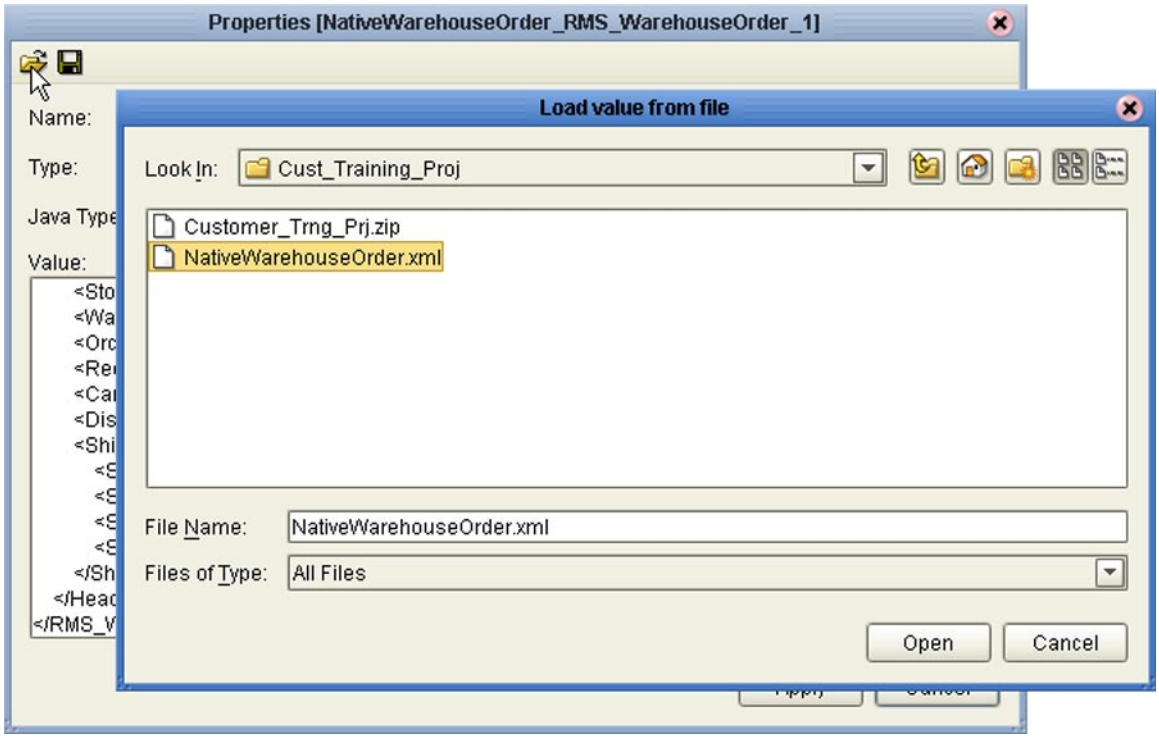
5 In the tester, right-click an OTD root node to display its context menu, and select **Properties** (see Figure 189).

Figure 189 Root Node Context Menu



6 In the *Properties* dialog box, invoke the File Browser (see Figure 190).

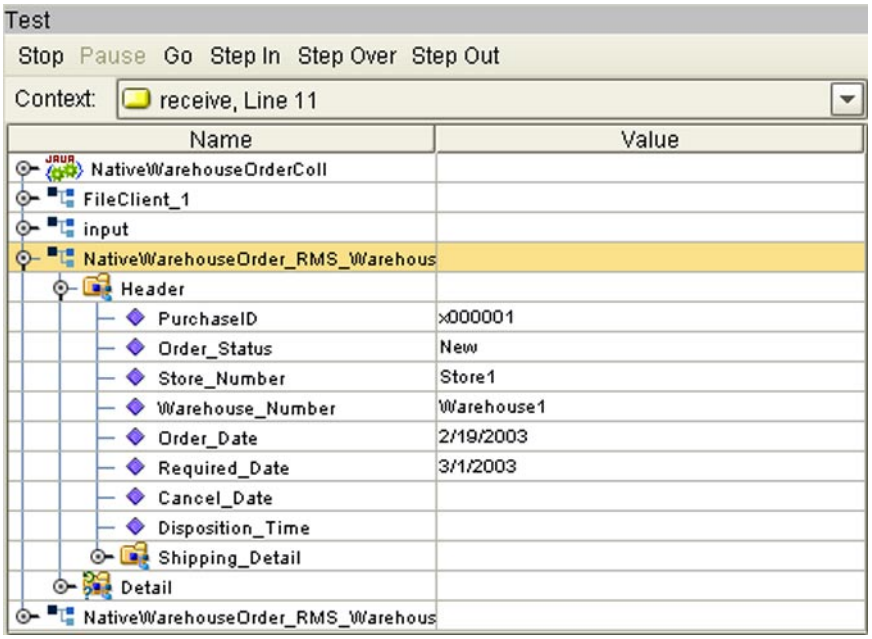
Figure 190 Selecting Input Data File



7 In the browser, select the desired input data file and click **Open**.

8 In the *Properties* dialog box, click **Apply**. The *Value* fields will be populated in the Test panel (see Figure 191).

Figure 191 Populated *Value* Fields



- Alternatively, you can manually type test data into the Value fields (double-quotes are not necessary for string values).

9 Click **Go** to run the tester through all the rules, or **Step Over** to test the rules one by one. (In this example all the rules are at a single level, so *Step In* and *Step Out* are not applicable.)

10 Click **Stop** to end the test.

11 Click **Start** to clear the Value fields and reset the Tester.

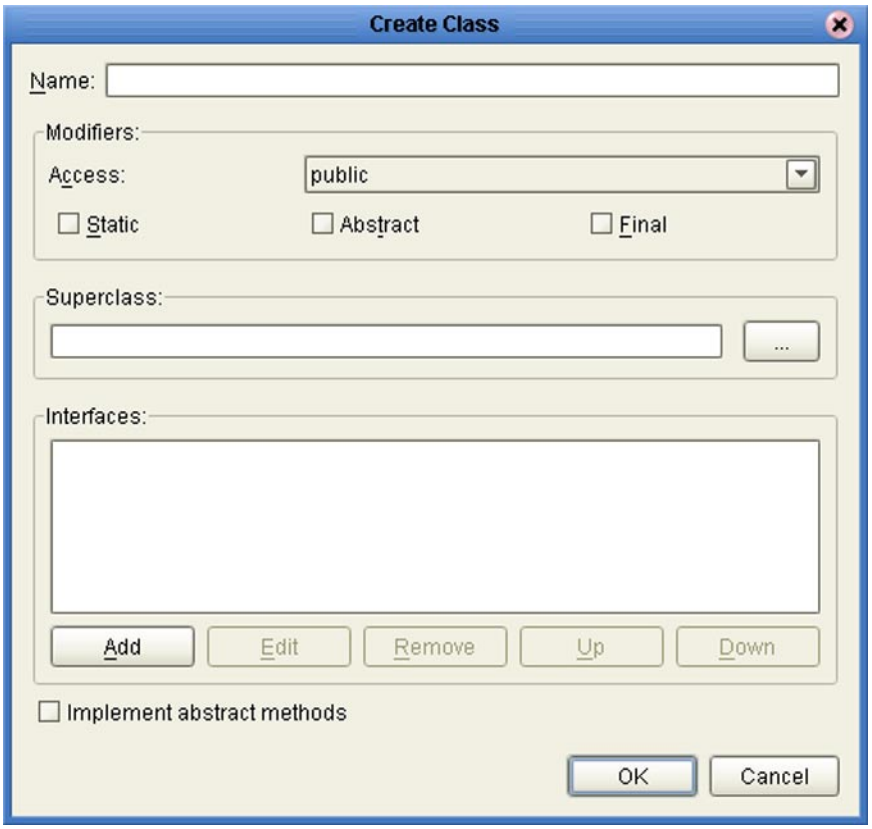
7.8 Design Procedures

7.8.1 Creating Classes

Clicking the **Class** icon in the Business Rules Editor toolbar (see Figure 192) displays the *Create Class* dialog box (see Figure 201).

Figure 192 Class Icon

Figure 193 Create Class Dialog Box



The various options are summarized in Table 68.

Table 68 Class Options

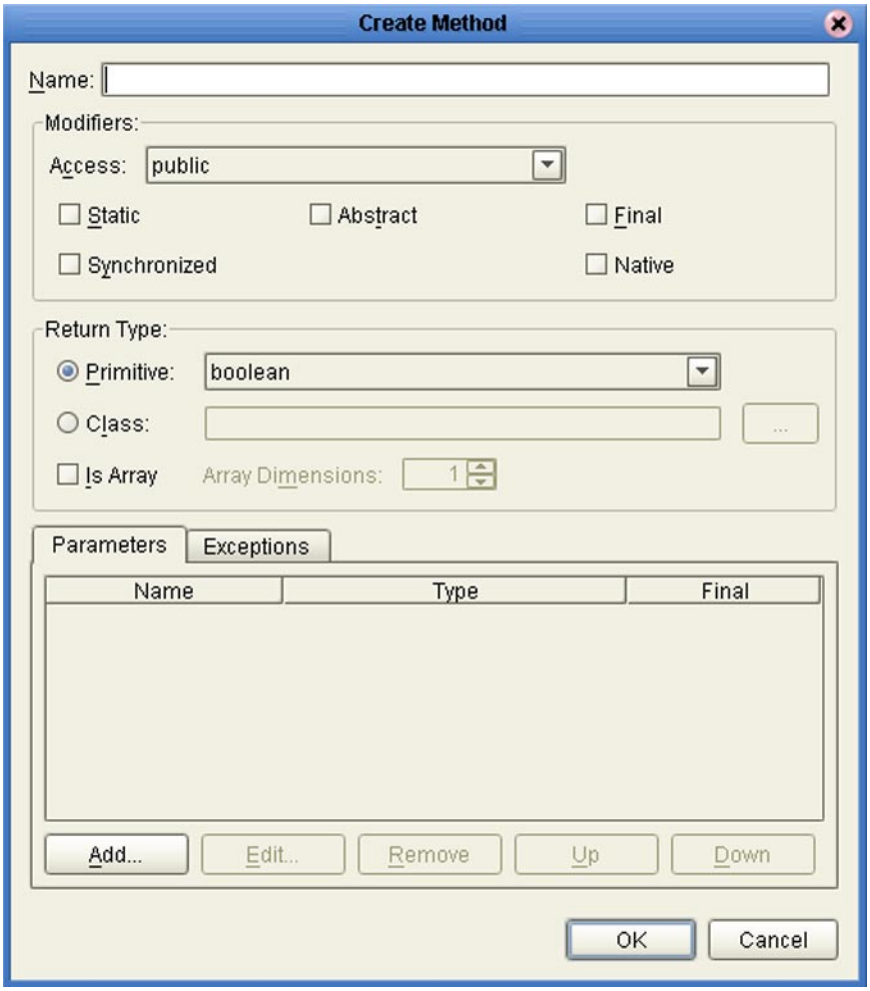
Heading	Option/Command	Description
Name		Your name for the class.
Modifiers	Access	One of the following: ▪ public ▪ protected ▪ private
	Static	Specifies that the class is static.
	Abstract	Specifies that the class is abstract.
	Final	Specifies that the class is final (no subclasses can be created).
Superclass		Assigns the class to another Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 227).
Interfaces		Displays a list of interfaces for the method.
	Add	Displays the Class Browser with which you can locate an interface to add (see **Using the Class Browser** on page 227).
	Edit	Displays a dialog box in which you can edit the interface.
	Remove	Deletes the selected interface.
	Up	Moves the selected interface up one level in the list.
	Down	Moves the selected interface down one level in the list.
Implement Abstract Methods		Permits the implementation of abstract methods within the class.

7.8.2 Creating Methods

Clicking the **Method** icon in the Business Rules Editor toolbar (see Figure 194) displays the *Create Method* dialog box (see Figure 195).

Figure 194 Method Icon

Figure 195 Create Method Dialog Box



The various options are summarized in Table 69.

Table 69 Method Options

Heading	Option/Command	Description
Name		Your name for the method.
Modifiers	Access	One of the following: ▪ public ▪ protected ▪ private
	Static	Specifies that the method is static.
	Synchronized	Specifies that the method is synchronized.
	Abstract	Specifies that the method is abstract.
	Final	Specifies that the method is final.
	Native	Specifies that the method is native.
Return Type	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the method to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 227).
	Is Array	Specifies that the method returns an array. When checked, enables the Array Dimensions list box, where you can set the array dimensions.
Parameters		Displays a list of parameters for the method.
	Add	Displays the Create Parameter dialog box, in which you can define a parameter (see **Creating Parameters** on page 252).
	Edit	Displays the Parameter dialog box (same as the Create Parameter dialog box), in which you can edit the selected parameter.
	Remove	Deletes the selected parameter.
	Up	Moves the selected parameter up one level in the list.
	Down	Moves the selected parameter down one level in the list.

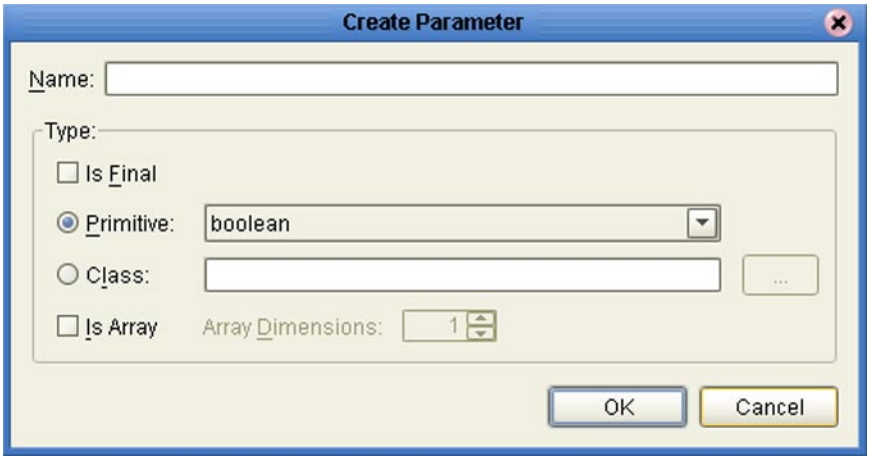
Table 69 Method Options

Heading	Option/Command	Description
Exceptions		Displays a list of exception classes thrown by the method.
	Add	Displays the Class Browser with which you can locate an exception class (see **Using the Class Browser** on page 227).
	Edit	Displays the Class Browser with which you can replace the selected class with another.
	Remove	Deletes the selected exception from the list.
	Up	Moves the selected exception class up one level in the list.
	Down	Moves the selected exception class down one level in the list.

Creating Parameters

Clicking **Add** in the Parameters panel of the *Create Method* dialog box displays the *Create Parameter* dialog box, shown in Figure 196.

Figure 196 Create Parameter Dialog Box



The various options are summarized in Table 70.

Table 70 Parameter Options

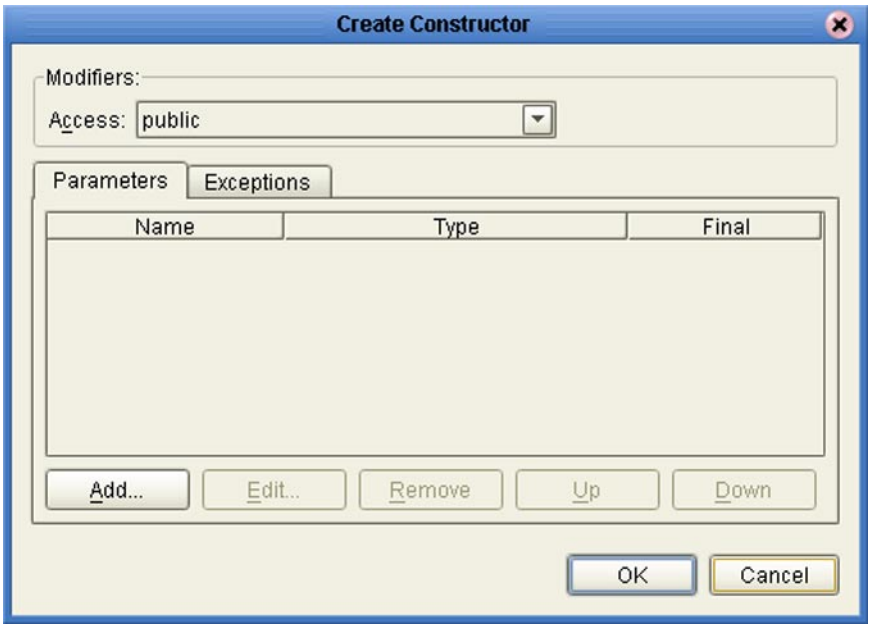
Heading	Option	Description
Name		Your name for the parameter.
Type	Is Final	Specifies that the parameter is final.
	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the parameter to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 227).
	Is Array	Specifies that the parameter is an array. When checked, enables the Array Dimensions field, where you set the array size (square).

7.8.3 Creating Constructors

Clicking the **Constructor** icon in the Business Rules Editor toolbar (see Figure 197) displays the *Create Constructor* dialog box (see Figure 201).

Figure 197 Constructor Icon

Figure 198 Create Constructor Dialog Box



The various options are summarized in Table 71.

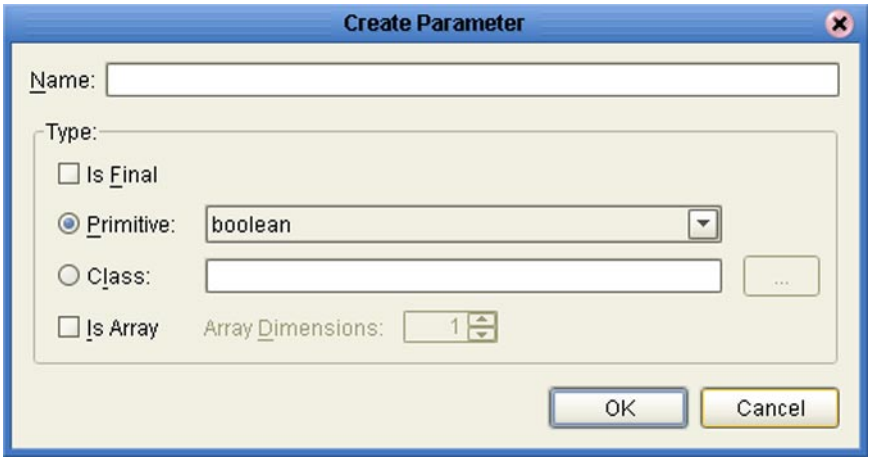
Table 71 Constructor Options

Heading	Option/Command	Description
Modifiers	Access	One of the following: ▪ public ▪ protected ▪ private
Parameters		Displays a list of parameters for the method.
	Add	Displays the Create Parameter dialog box, in which you can define a parameter (see **Creating Parameters** on page 256).
	Edit	Displays the Parameter dialog box (same as the Create Parameter dialog box), in which you can edit the selected parameter.
	Remove	Deletes the selected parameter.
	Up	Moves the selected parameter up one level in the list.
	Down	Moves the selected parameter down one level in the list.
Exceptions		Displays a list of exception classes thrown by the method.
	Add	Displays the Class Browser with which you can locate an exception class (see **Using the Class Browser** on page 227).
	Edit	Displays the Class Browser with which you can replace the selected class with another.
	Remove	Deletes the selected exception from the list.
	Up	Moves the selected exception class up one level in the list.
	Down	Moves the selected exception class down one level in the list.

Creating Parameters

Clicking **Add** in the Parameters panel of the *Create Constructor* dialog box displays the *Create Parameter* dialog box, shown in Figure 196.

Figure 199 Create Parameter Dialog Box



The various options are summarized in Table 70.

Table 72 Parameter Options

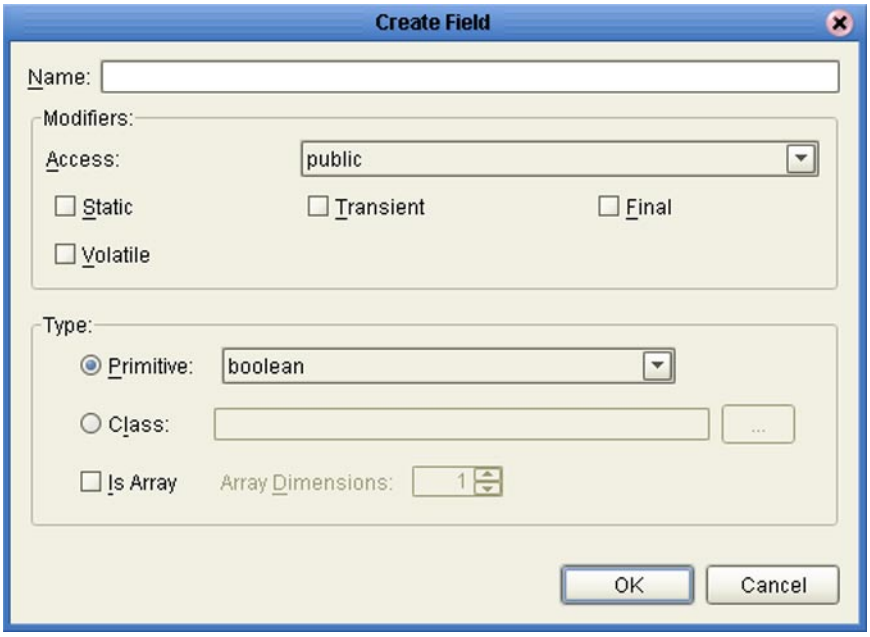
Heading	Option	Description
Name		Your name for the parameter.
Type	Is Final	Specifies that the parameter is final.
	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the parameter to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 227).
	Is Array	Specifies that the parameter is an array. When checked, enables the Array Dimensions field, where you set the array size (square).

7.8.4 Creating Fields

Clicking the **Field** icon in the Business Rules Editor toolbar (see Figure 200) displays the **Create Field** dialog box (see Figure 201).

Figure 200 Field Icon

Figure 201 Create Field Dialog Box



The various options are summarized in Table 73.

Table 73 Field Options

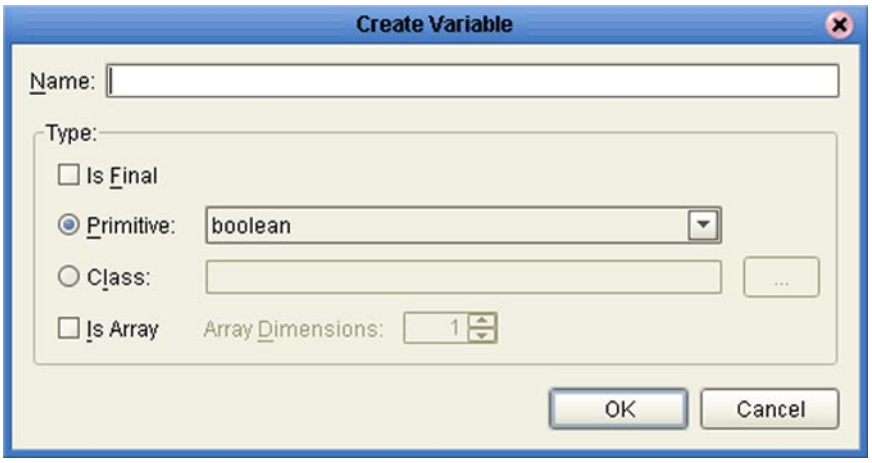
Heading	Option	Description
Name		Your name for the field.
Modifiers	Access	One of the following: ▪ public ▪ protected ▪ private
	Static	Specifies that the field is static.
	Volatile	Specifies that the field is volatile.
	Transient	Specifies that the field is transient.
	Final	Specifies that the field is final.
Type	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the field to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 227).
	Is Array	Specifies that the variable is an array. When checked, enables the Array Dimensions list box, where you can set the array dimensions.

7.8.5 Creating and Modifying Variables

Clicking the **Local Variable** icon in the Business Rules Editor toolbar (see Figure 202) displays the **Create Variable** dialog box (see Figure 203).

Figure 202 Local Variable Icon

Figure 203 Create Variable Dialog Box



The various options are summarized in Table 74.

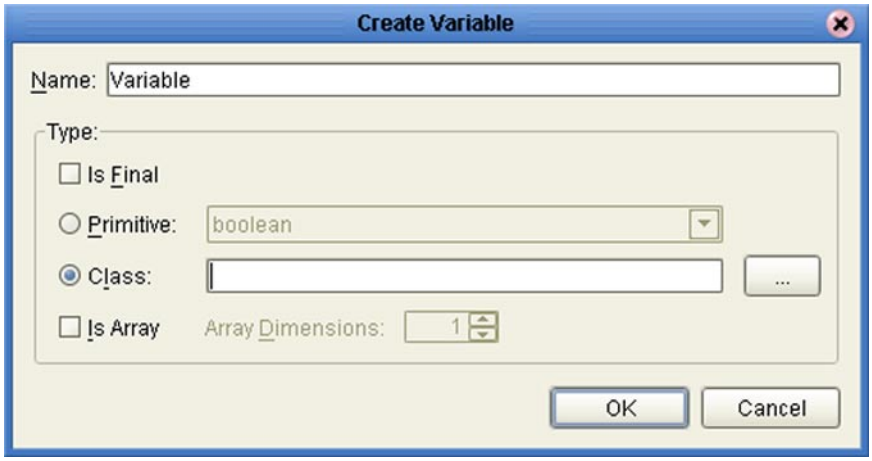
Table 74 Variable Options

Heading	Option	Description
Variable Name		Your name for the variable.
Type	Is Final	Specifies that the variable is final.
	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the variable to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 227).
	Is Array	Specifies that the variable is an array. When checked, enables the Array Dimensions list box, where you can set the array dimensions.

To create a variable of a specific class type

1 In the Business Rules editor, click the **Local Variable** button to display the dialog box shown in Figure 204.

Figure 204 Create Variable Dialog Box

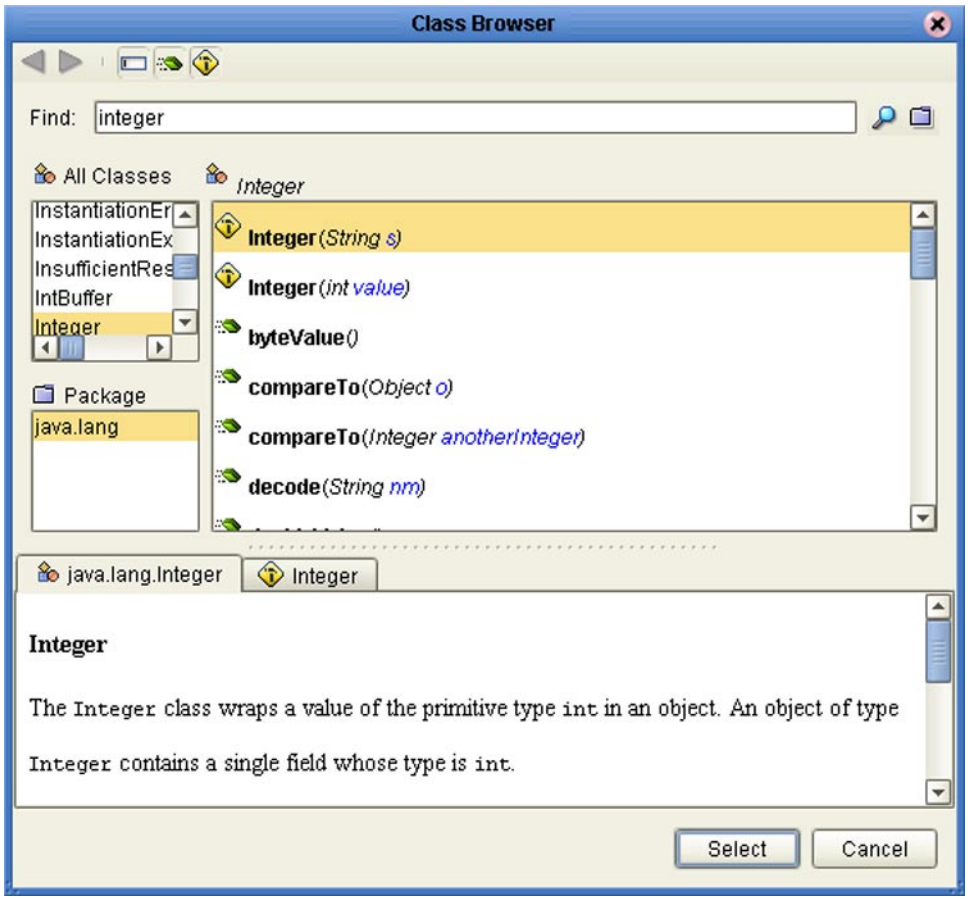


2 Enter a name for the variable in the **Name** text box.

3 Select the **Class** option button.

4 Click the **Ellipsis** (**…**) button to display the **Class Browser** dialog box shown in Figure 205.

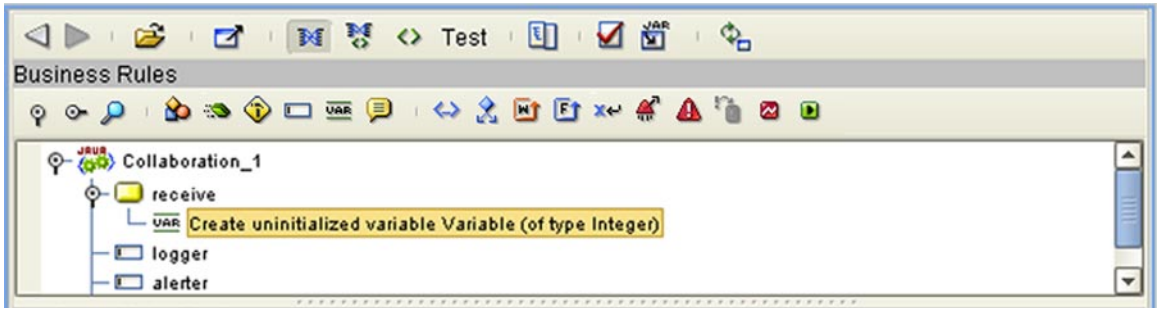
Figure 205 Class Browser - Select Class



5 Select a class from the **All Classes** list (for example, *Integer*).

6 Click the **Select** button to add the variable to the Business Rules editor as shown in Figure 206.

Figure 206 Business Rules with Variable



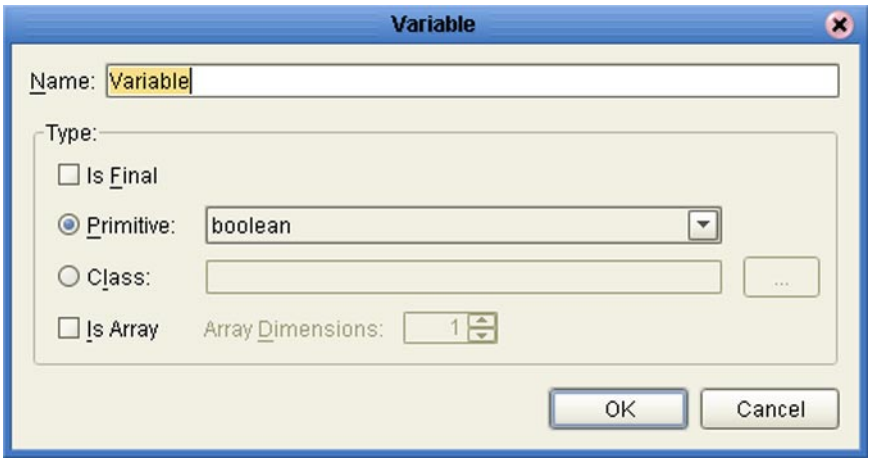
Note: *Having the variable selected when the constructor is created initializes the variable. This also occurs when a literal is created, for those variables that can have literals assigned.*

To modify an existing variable

1 Right-click on the variable in the OTD tree to display its context menu.

2 Select **Open Declaration** to display the *Variable* dialog box (see Figure 207).

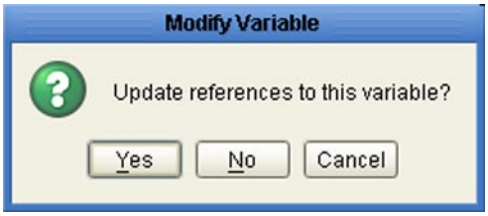
Figure 207 Variable Dialog Box



3 Make the desired changes to the variable properties.

4 Click **OK** to display the dialog box shown in Figure 208.

Figure 208 Modify Variable Dialog Box



5 Click the appropriate button, as follows:

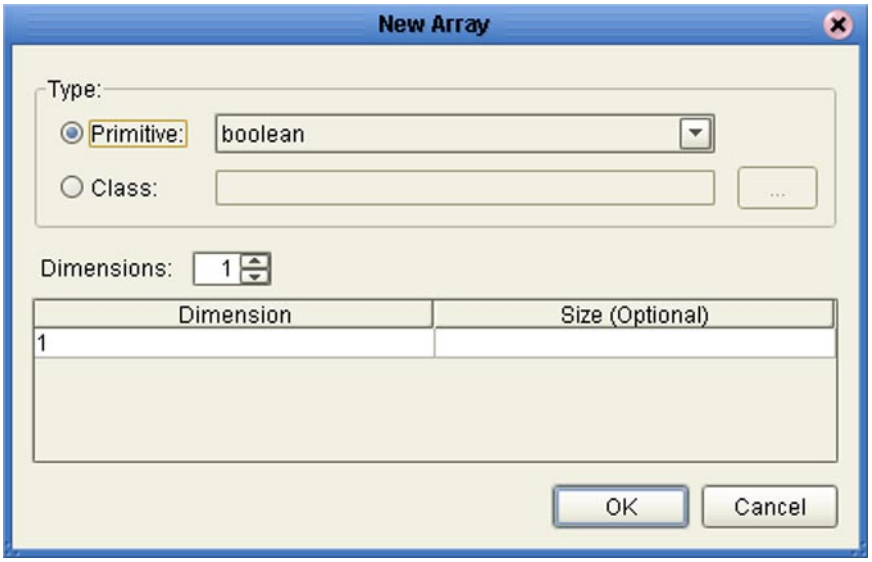
- **Yes** to save your modifications and update all references to the variable.
- **No** to save your modifications without updating any references to the variable.
- **Cancel** to return to the *Variable* dialog box to change or cancel your modifications.

7.8.6 Creating Arrays

Clicking the **New Array** icon in the Business Rules Designer toolbar (see Figure 209) displays the **New Array** dialog box (see Figure 210).

Figure 209 New Array Icon

Figure 210 New Array Dialog Box



The various options are summarized in Table 75.

Table 75 Array Options

Heading	Option	Description
Array Type	Primitive	One of the following: ▪ boolean ▪ byte ▪ char ▪ short ▪ int ▪ long ▪ float ▪ double
	Class	Assigns the field to a Java class. Clicking the ellipsis (**...**) button launches the *Class Browser* (see **Using the Class Browser** on page 227).

Table 75 Array Options

Heading	Option	Description
Dimensions		You can set a dimension value for the array using the list box.
	Dimension	
	Size	

7.8.7 Creating and Modifying Literals

Using the Toolbar Menus

The Business Rules Designer toolbar menus contain several literal methods, listed in Table 76, which are described in **Predefined Collaboration Methods (Java)** on page 210. Clicking the menu option places the corresponding method box on the mapping canvas.

Table 76 Literals in Toolbar Menus

Menu	Literal
Boolean	False
	True
Math	Literal Number
Object	Null
String	Literal Character
	Literal String

Using the Method Dialog Box

Launch the method dialog box as described in **Using the Method Dialog Box** on page 232 and click the **Create Literal** icon in the toolbar. If you have launched the dialog box from an Object, the icon depicts an Object (see Figure 211) and the **Null** method box is placed on the mapping canvas.

Figure 211 Object Icon

If you have launched the icon from a String, the icon depicts a String (seeFigure 212) and the **Literal String** method box is placed on the mapping canvas.

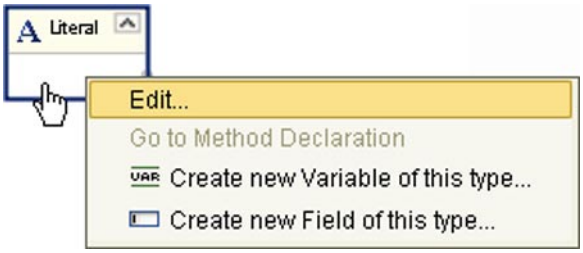
Figure 212 String Icon

Editing a Literal

You can edit a literal by:

- Double-clicking the value field in the method box, or
- Right-clicking the value field to display the menu shown in Figure 213 and selecting **Edit…**.

Figure 213 Literal Menu



Either procedure enables the value field for editing. In the case of a Boolean, you can switch between **true** and **false**. In the case of a character or string, you can type text into the field.

Note: *Use **Ctrl-Enter** to insert a line break.*

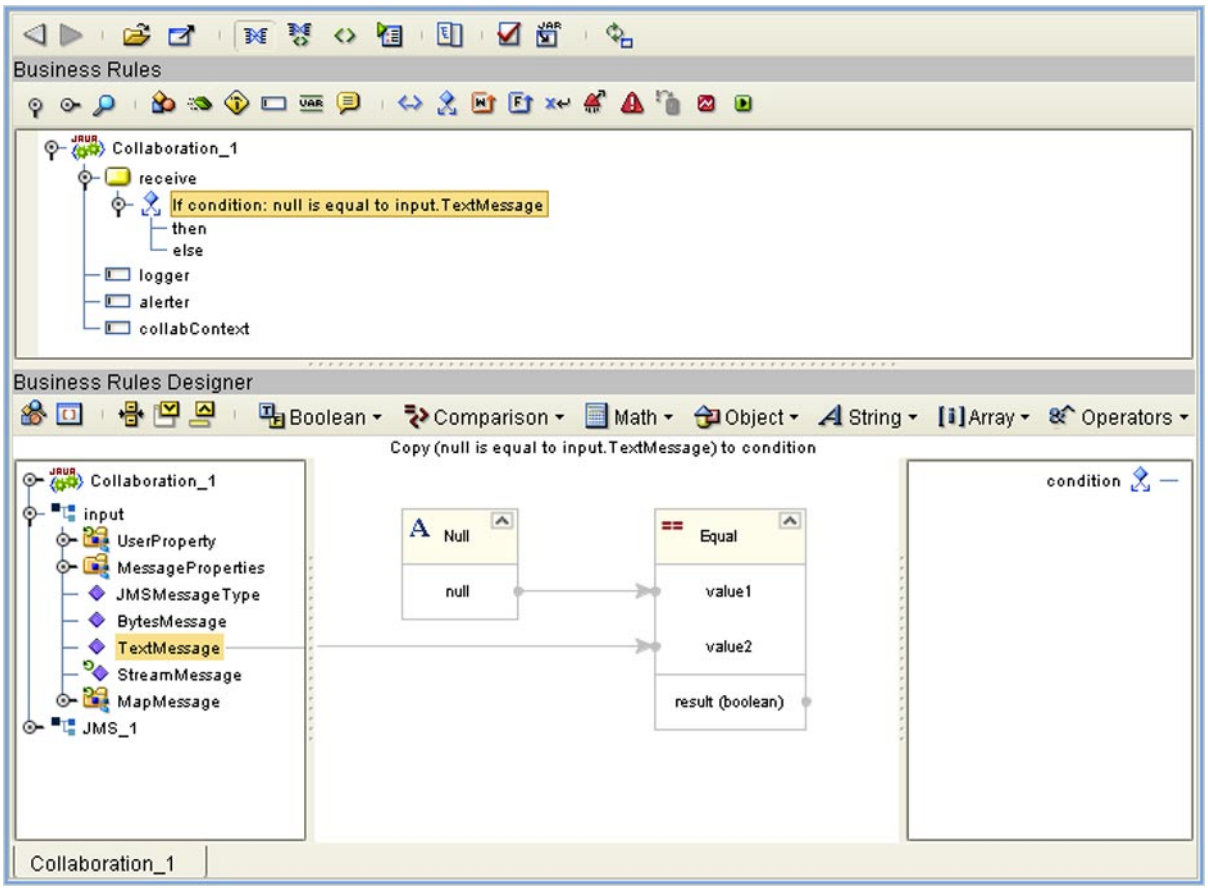
7.8.8 Generating Alerts

You can incorporate alert generation into a Collaboration Definition (Java). As an example, you could perform a test to determine whether or not a file is empty, and raise an alert if it is. The following procedure illustrates this example.

To create an alert using the Collaboration Editor (Java)

1. Add an **If** condition to the appropriate method in the Business Rules editor (see Figure 214).
2. Define the rule in the Business Rules Designer by placing an **equal** method box (*Comparison* menu) and a **literal null** method box (*Object* menu) in the mapping pane.
3. Link the desired input file to one input port of the **equal** method box, and the literal output port to the other **equal** input port., as shown.

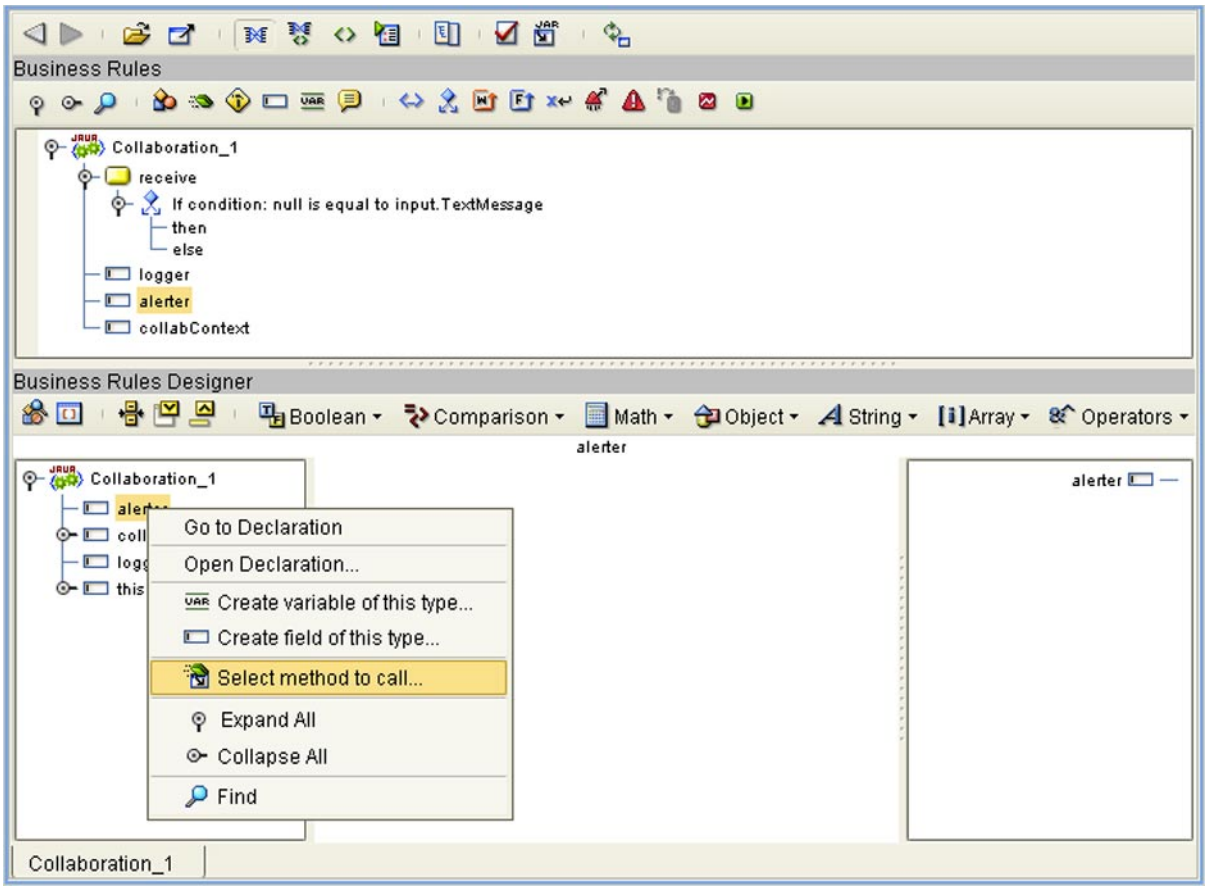
Figure 214 Empty File Test - If Condition



4 Initiate an alert object by clicking the **alerter** node of your Collaboration Definition in the Business Rules editor.

5 Right-click the **alerter** node of the Collaboration Definition in the left pane of the Business Rules Designer, which displays the menu shown in Figure 215.

Figure 215 Empty File Test - Alerter Node Menu

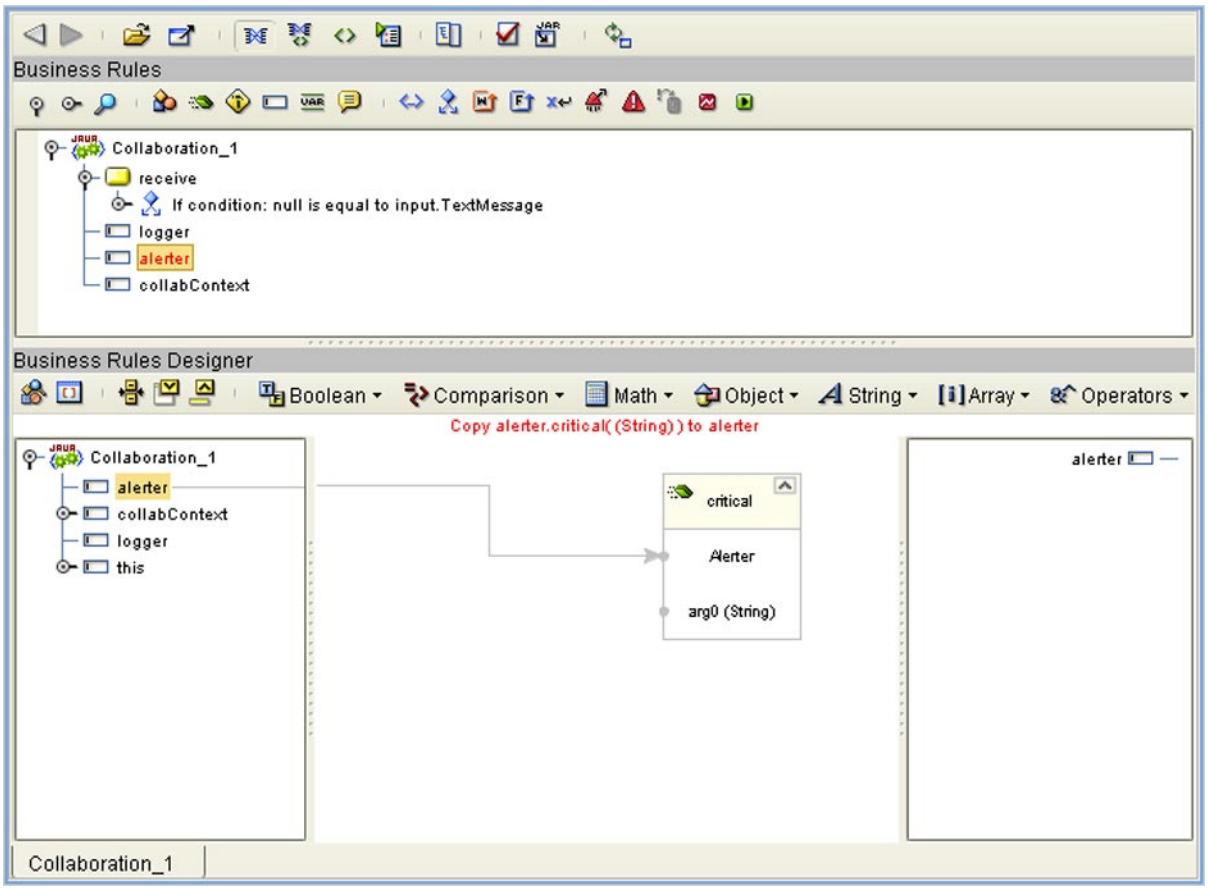


6 Click **Select a method to call…**, which displays a method list box containing severity levels (see Figure 216).

Figure 216 Alerter Severity Method List

7 Select the desired severity for the alert from the method list box; for example, **critical**. A **critical** method box is placed in the mapping panel (see Figure 217).

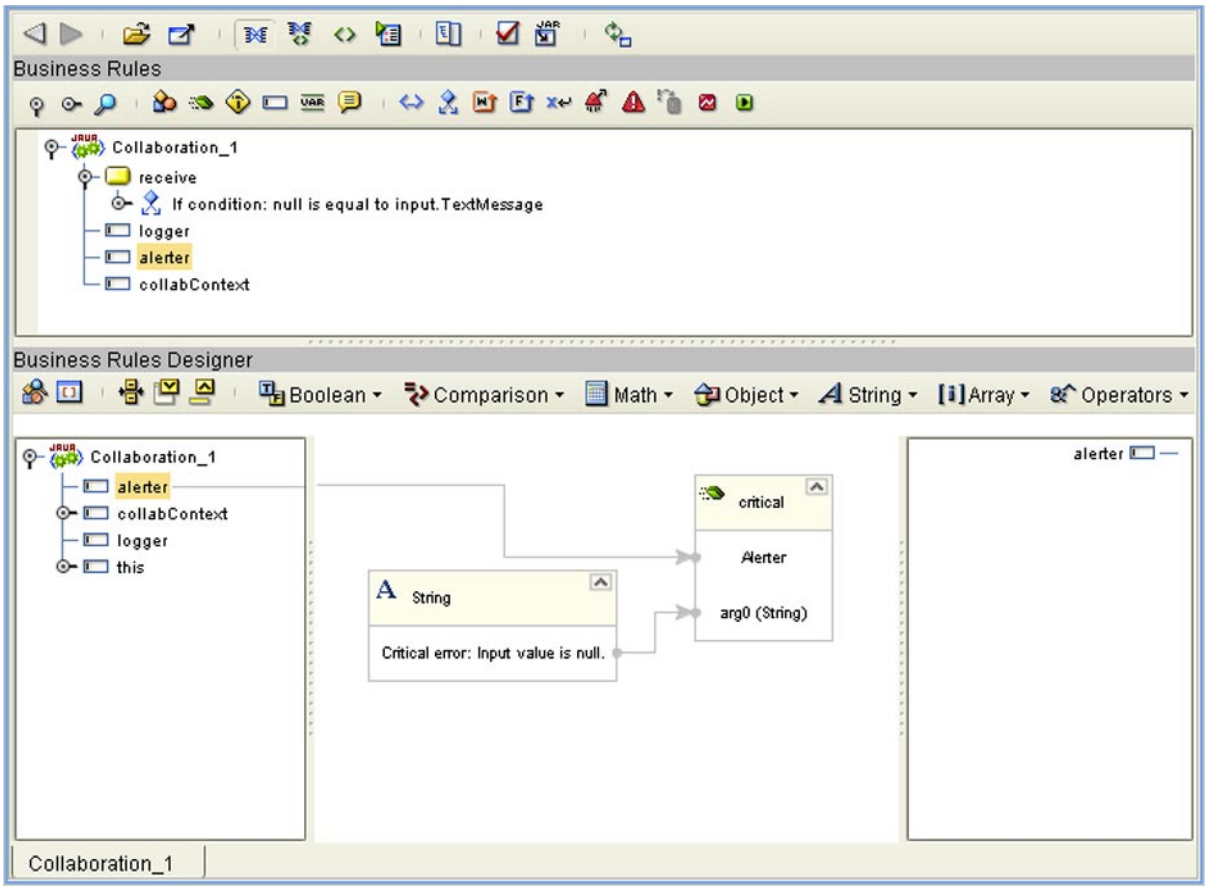
Figure 217 Empty File Test - *critical* Method



8 Create your alert message, which can be a literal, a constant, or an OTD field name. As an example, we show a **literal string**.

9 Pass the alert message to the alert event by linking the literal message output port to the argument of the alerter object (see Figure 218).

Figure 218 Empty File Test - Pass Alert Message



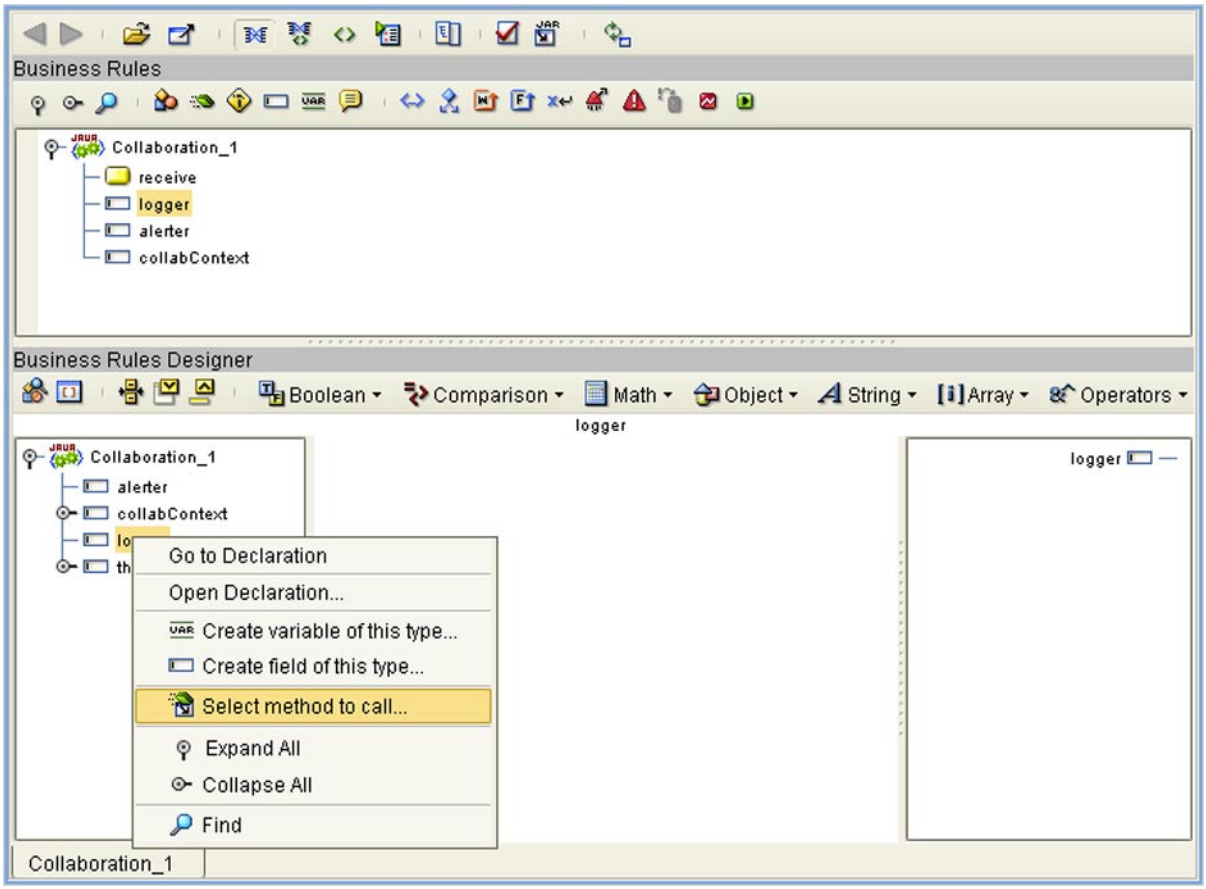
7.8.9 Creating Log Entries

You can initiate log entries from a Collaboration Definition (Java), as described in the following procedure.

To create a log entry using the Collaboration Editor (Java)

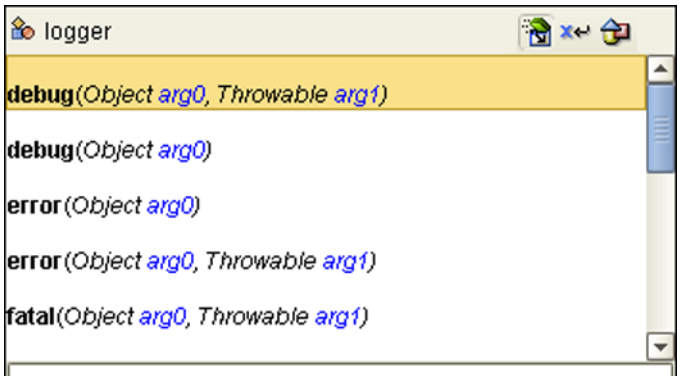
1. Initiate a logging event by clicking the **logger** node of your Collaboration Definition in the Business Rules editor.
2. Right-click the **logger** node for the Collaboration Definition in the left pane of the Business Rules Designer, which displays the menu shown in Figure 219.

Figure 219 Logging Example - Logger Node Menu



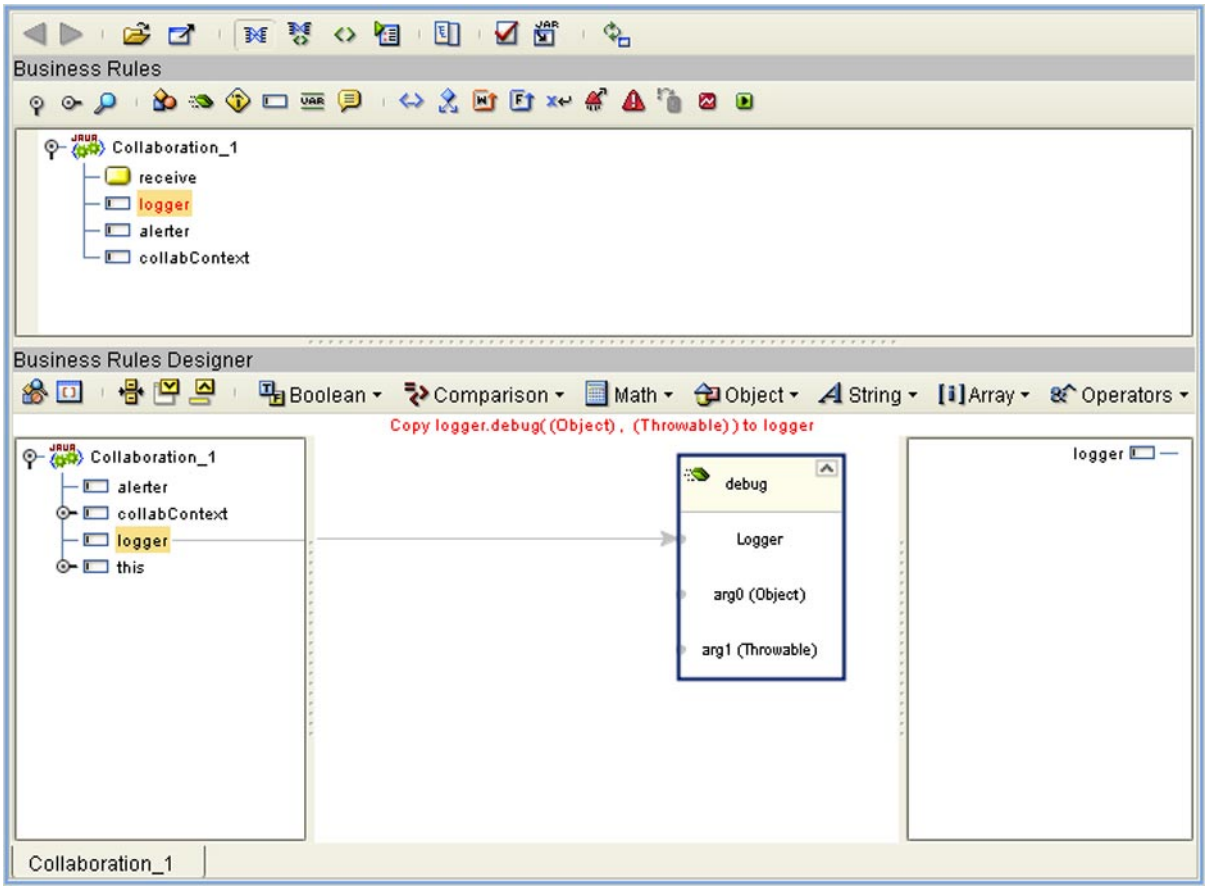
3. Click **Select a method to call…**, which displays a method list box containing logging levels (see Figure 220).

Figure 220 Logging Level Method List



4 Select the desired logging level from the method list box; for example, **debug**. A **debug** method box is placed in the mapping panel (see Figure 221).

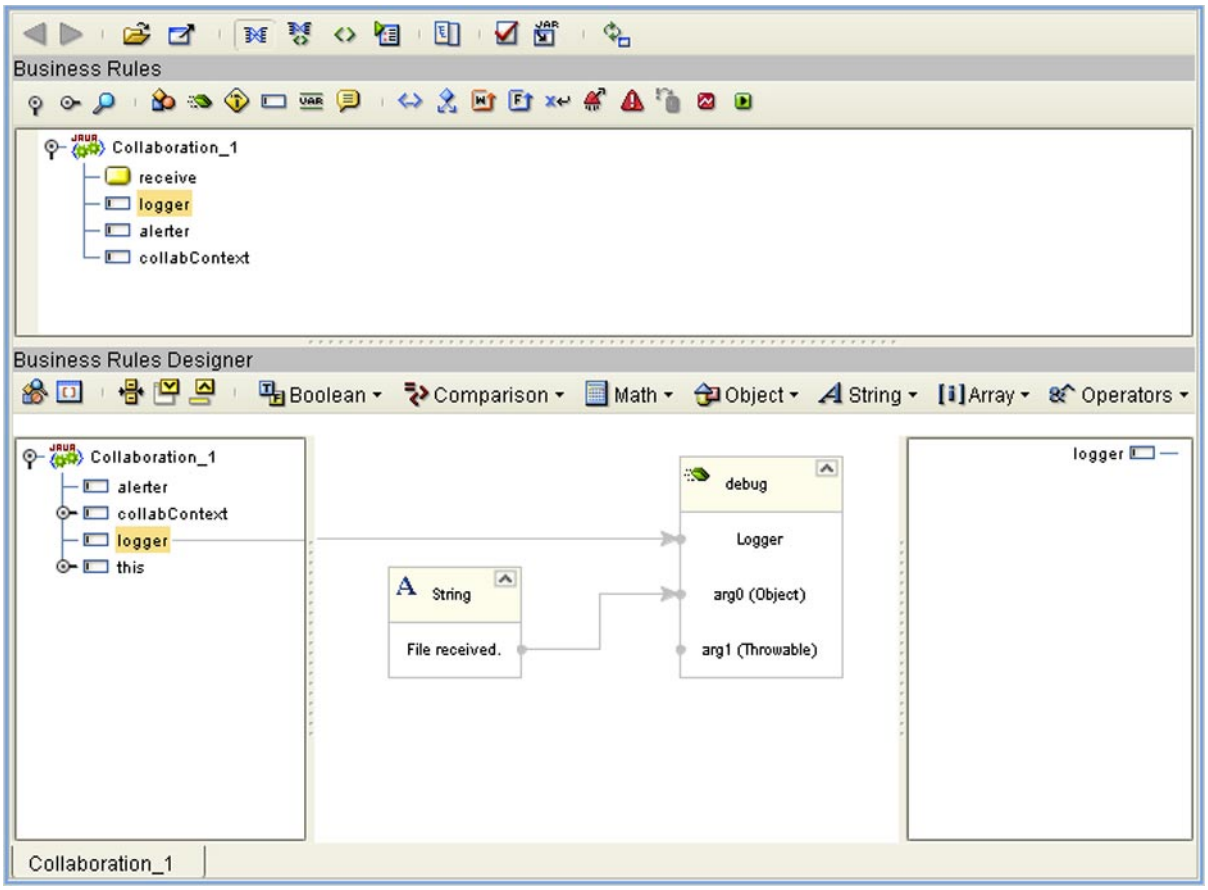
Figure 221 Logging Example - *debug* Method Box



5 Create your log message, which can be a literal, a constant, or an OTD field name. As an example, we show a **literal string**.

6 Pass the log message to the logging event by dragging the message to the appropriate argument provided by the method box (see Figure 222).

Figure 222 Logging Example - Pass Log Message



Note: *Compiler errors are logged in the IDE log (\edesigner\usrdir\system\IDE.log), at the **warning** level. See the eGate Integrator System Administration Guide for more information regarding logging level settings.*

7.8.10 Importing Third-Party Java Classes

Adding third-party Java classes (**.jar** files) to a Collaboration Definition is a two-step process:

1. First, you must import the Java classes into your Project.
2. Second, you must add the Java classes to the specific Collaboration Definition.

To import third-party Java classes into a Project

1. Copy the third-party files to a convenient directory.
2. Right-click on a **Project** in the Project Explorer to display its context menu.
3. Select **New > File…** in the context menu (see Figure 223) to display the file import dialog box (see Figure 224).

Figure 223 Project Context Menu: New File

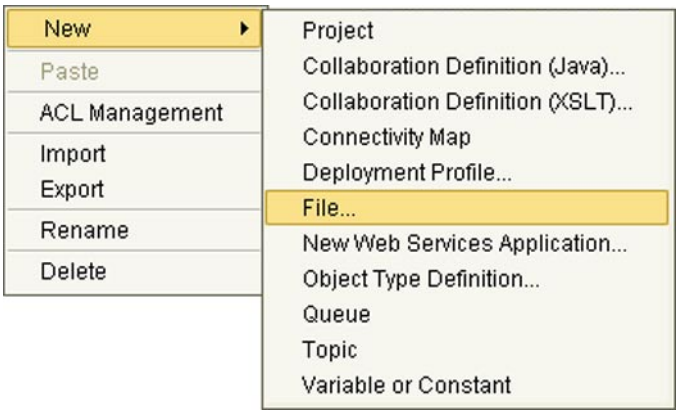
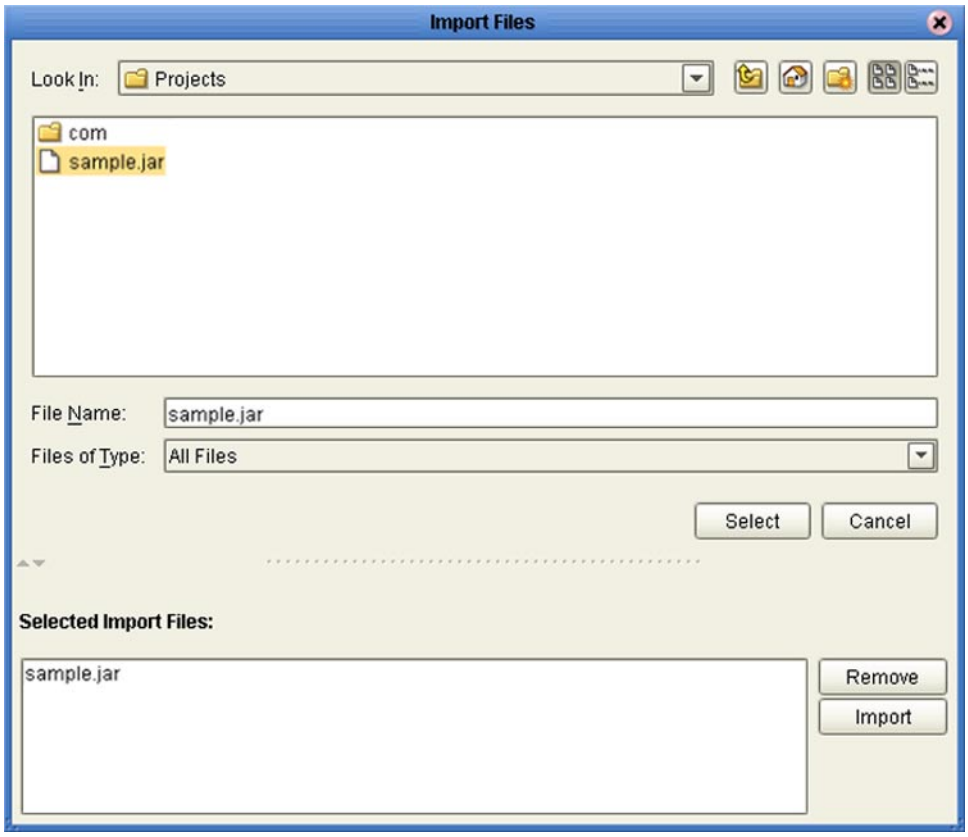


Figure 224 File Import Dialog Box



4 Locate the desired **.jar** files and click **Select** for each.

5 When you have selected all desired files, click **Import** to import the entire list into the Project.

6 Add the Java classes to your Collaboration Definition following the procedure described in **Adding Java classes to Collaboration Definitions** on page 276.

7.8.11 Adding Java classes to Collaboration Definitions

You can use the Collaboration Editor to add Java classes to your Collaboration Definitions. These Java classes can be from within the current Project, or from another Project (such as the ICAN Project template, for example).

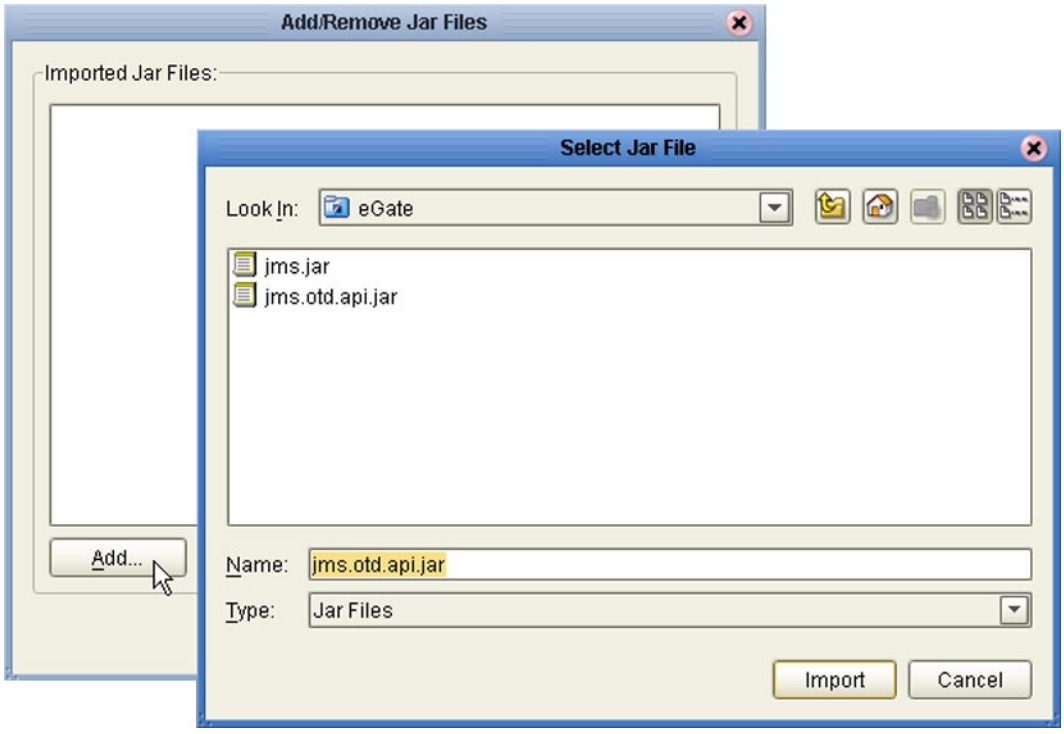
To add Java classes to a Collaboration Definition:

1. Open the Collaboration to which you want to add Java classes.
2. Click the **JAR File** icon in the main Collaboration Editor toolbar (see Figure 225), which displays the *Add/Remove Jar Files* dialog box.

Figure 225 JAR File Icon

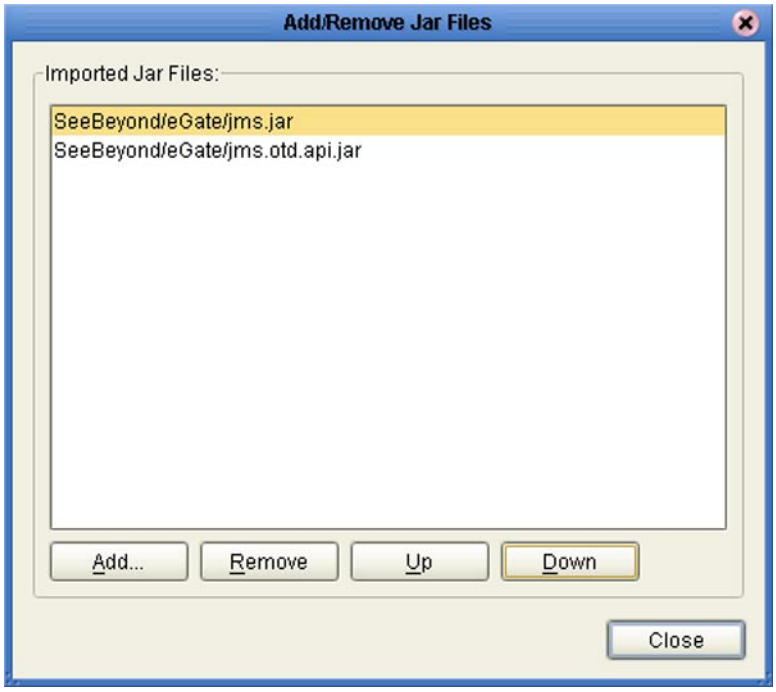
3. Click **Add** in the dialog box to display a file browser box (see Figure 226).
4. Locate the desired **.jar** files and click **Import** for each.

Figure 226 Add/Remove Jar File Dialog Box (1)



5 When all the **.jar** files are listed in the *Add/Remove Jar Files* dialog box (see Figure 227), click **Close**.

Figure 227 Add/Remove Jar File Dialog Box (2)



- You can also remove **.jar** files by selecting the files and clicking **Remove**, and change the order of the files by using the **Up** and **Down** buttons (files are searched in order, as in the *classpath*).

7.8.12 Using *Try-Catch*

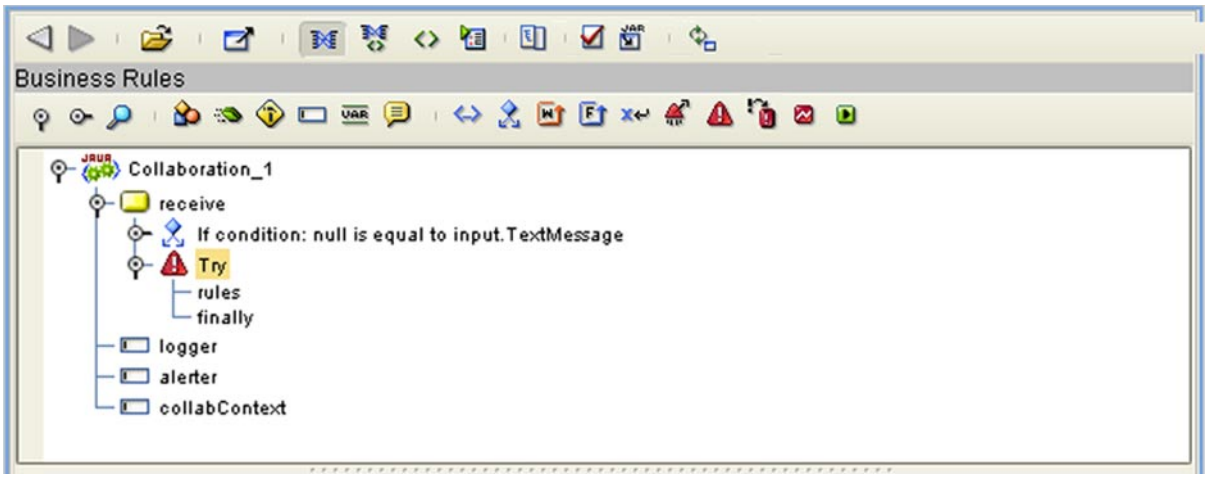
Clicking the **try** icon in the toolbar adds a **try** statement to the Business Rules tree, initiating a number of programming statements that are monitored to see whether they succeed or fail. A **finally** statement is added automatically. To perform a **try-catch**, you must add the **catch** statement manually as described in the following procedure.

To create a try-catch operation

1 Click the **Try** icon in the toolbar (see Figure 228), which adds the **try** statement to the business rules tree (see Figure 229). It also enables the **Catch** icon in the Business Rules editor toolbar.

Figure 228 Try Icon

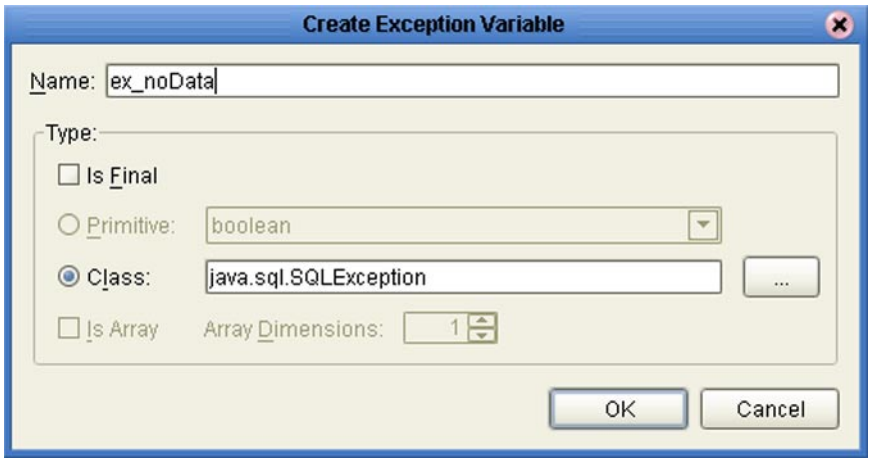
Figure 229 Try Statement



2 Select the **Catch** icon (see Figure 230), which displays the **Create Exception Variable** dialog box (see Figure 231).

Figure 230 Catch Icon

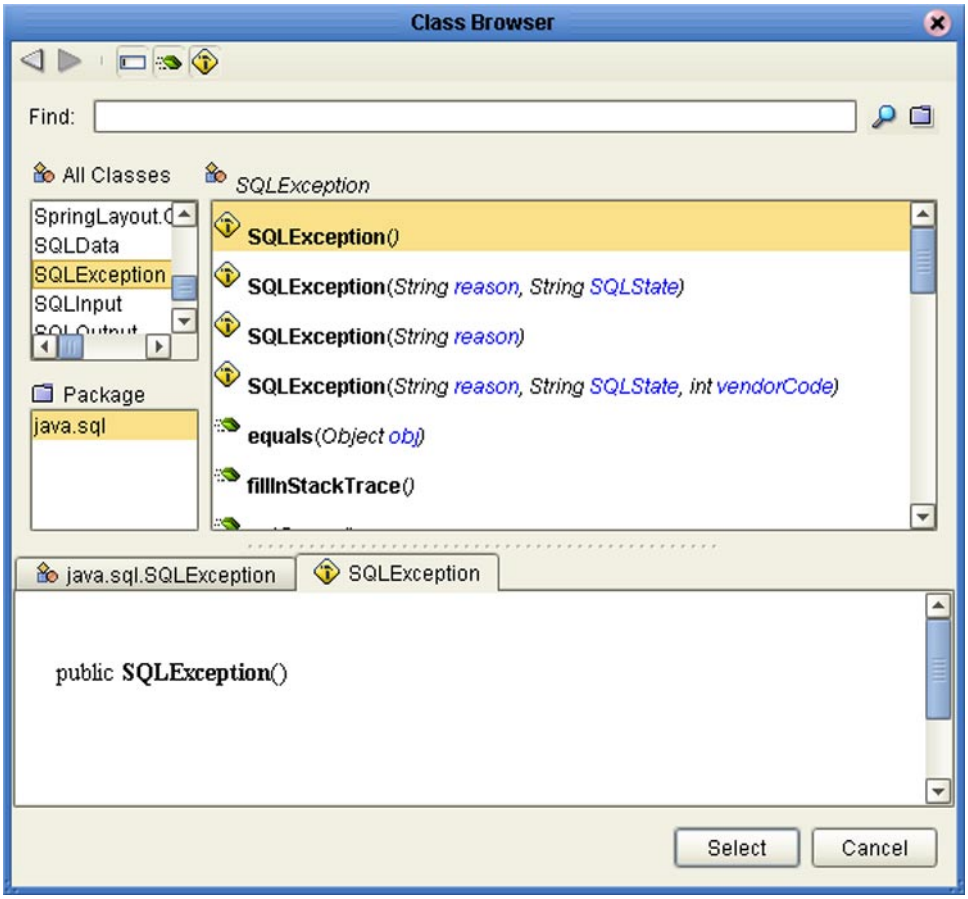
Figure 231 Create Exception Variable Dialog Box



3 Enter a name for the exception, for example: **ex_noData**.

4 Click the **Ellipsis** (**…**) button to display the **Class Browser** dialog box shown in Figure 232.

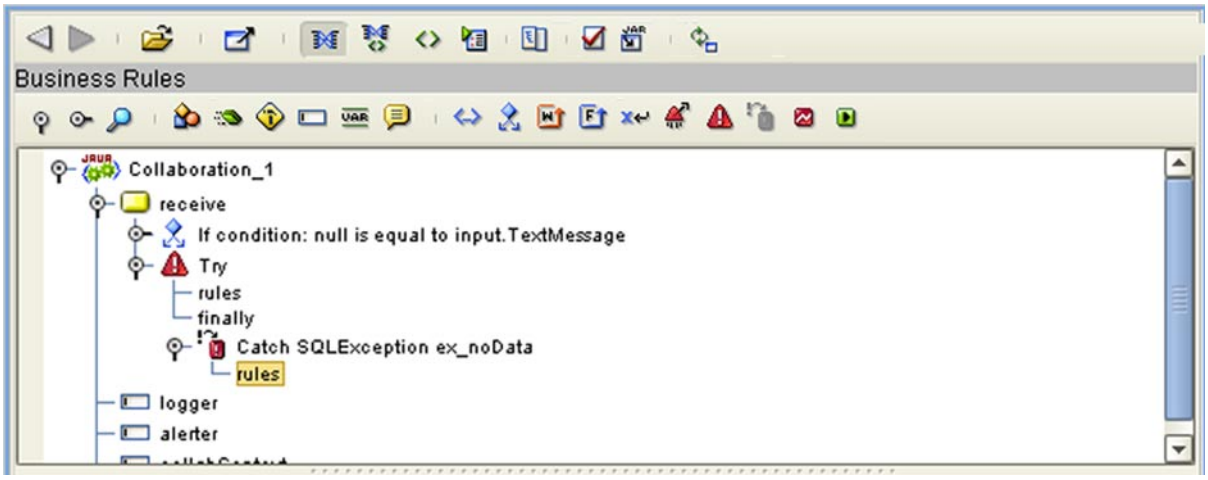
Figure 232 Class Browser Dialog Box



5 To specify that the exception represents a database error, for example, select **SQLException** and click **Select**.

6 The **catch** statement is now added to the business rules tree (see Figure 233).

Figure 233 Catch Statement



7 The rule shown in Figure 233 can contain a literal representing the message that will be assigned to an outbound element when the rule is executed.

7.8.13 Using Byte-Array Converters

Note: *Byte-array converters are currently available only when using either Japanese or Korean localized versions of eGate Integrator.*

Several data-encoding converters are available for byte arrays (byte[]s) within eGate Integrator. To access these converters, you must enable the extended language options as described in **Options Setup** on page 57.

Table 77 Byte-Array Encoding Converters - Japanese

Converter Designation	Converts	
	From	To
SJIS->EBCDIC-J	SJIS	EBCDIC-J
SJIS->JEF	SJIS	JEF
SJIS->JIPSE	SJIS	JIPSE
SJIS->KEIS	SJIS	KEIS
EBCDIC-J->SJIS	EBCDIC-J	SJIS
JEF->SJIS	JEF	SJIS
JIPSE->SJIS	JIPSE	SJIS
KEIS->SJIS	KEIS	SJIS

Table 78 Byte-Array Encoding Converters - Korean

Converter Designation	Converts	
	From	To
EUC-KR->EBCDIC-K	EUC-KR	EBCDIC-K
EBCDIC-K->EUC-KR	EBCDIC-K	EUC-KR

Defining Encoding Converter Methods

Encoding converter methods are defined using a dialog box, shown in Figure 235, that is invoked by clicking the **Encoding Converter** icon in the Business Rules Designer toolbar (see Figure 234). The options in the dialog box are described in Table 79.

Figure 234 Encoding Converter Icon

Figure 235 Encoding Converter Methods Dialog Box

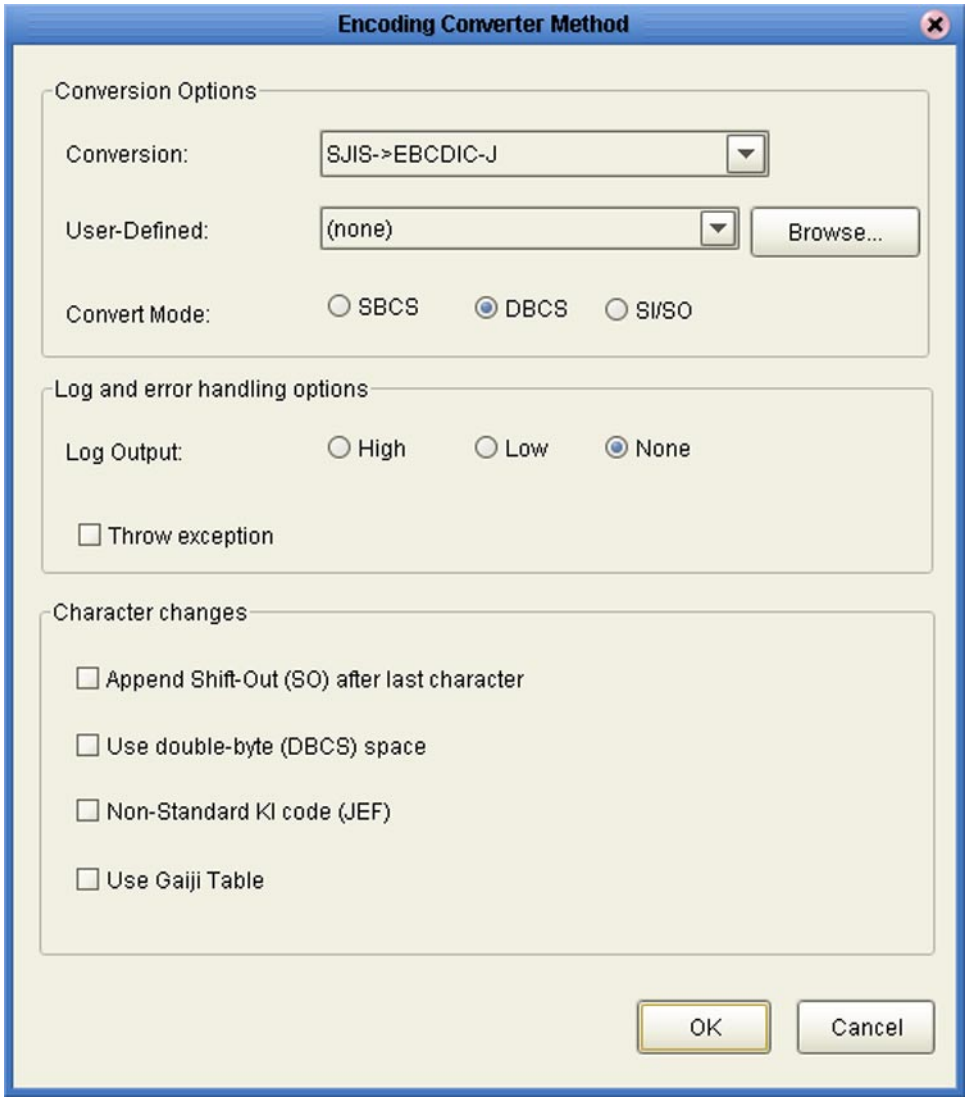


Table 79 Encoding Converter Methods Dialog Box Options

Option	Description
Conversion	Specifies the type of byte-array encoding converter to use (see Table 77 and Table 78 for list of converters). Applies to all converters.
User-Defined	Specifies the location of a user-defined mapping table, in which exceptions to the standard mapping can be entered. This mapping table is checked first when performing a conversion, and its contents will override the conversion logic of the standard conversion specified above. If the specific character is not found in this table, then the standard conversion mapping will be used. The format of this table is an ASCII text file with the following characteristics: ▪ # = comment. ▪ one line = one record. ▪ one record contains a source code point in HEX format, followed by a blank space, followed by a destination code point in HEX format. ▪ HEX format can be with or without **0x**. Applies to all converters except those for EBCDIC-J and EBCDIC-KR.
Convert Mode	Specifies whether the field is SBCS only, DBCS only (with no SI/SO characters), or mixed with SI/SO characters. Applies to all converters.
Log Output	Specifies the logging level to use during conversion. Applies to all converters except those for EBCDIC-J and EBCDIC-KR.
Throw Exception	If checked, converter will throw an exception when characters are out of range of SJIS. If unchecked, it will convert to a question mark (**?**), consistent with the Java specification. Applies to all converters except those for EBCDIC-J and EBCDIC-KR.
Append Shift-Out after last character	If checked, converter appends the last SO character when the data ends with a double byte character. Applies to all SJIS->xxx converters except SJIS->EBCDIC-J when the SI/SO option is selected for Convert Mode.
Use double-byte (DBCS) space	If checked, converts two single byte spaces into one double byte space. Applies to all xxx->SJIS converters except EBCDIC-J->SJIS when the DBCS option is selected for Convert Mode.

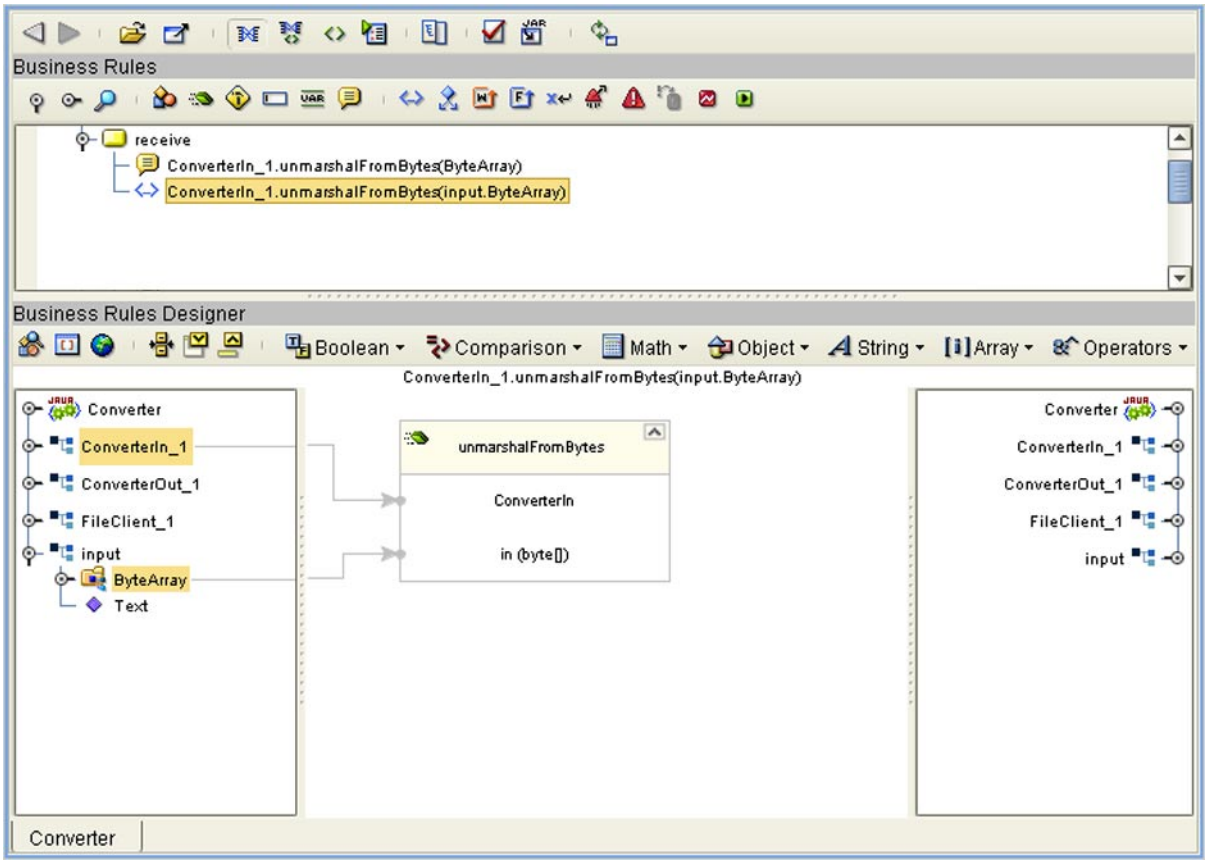
Table 79 Encoding Converter Methods Dialog Box Options

Option	Description
Non-Standard KI code (JEF)	If checked, uses 0x38 instead of standard 0x28 as the KI code. Applies only to JEF converters when the SI/SO option is selected for Convert Mode.
Use Gaiji Table	If checked, converter will use the User-Defined mapping table specified above. Applies to all converters except those for EBCDIC-J and EBCDIC-KR.

To use a byte-array encoding converter in a Collaboration

1 Define you business rules to receive the input data as a byte array, using the **unmarshalFromBytes** Java method mapped as shown in Figure 236.

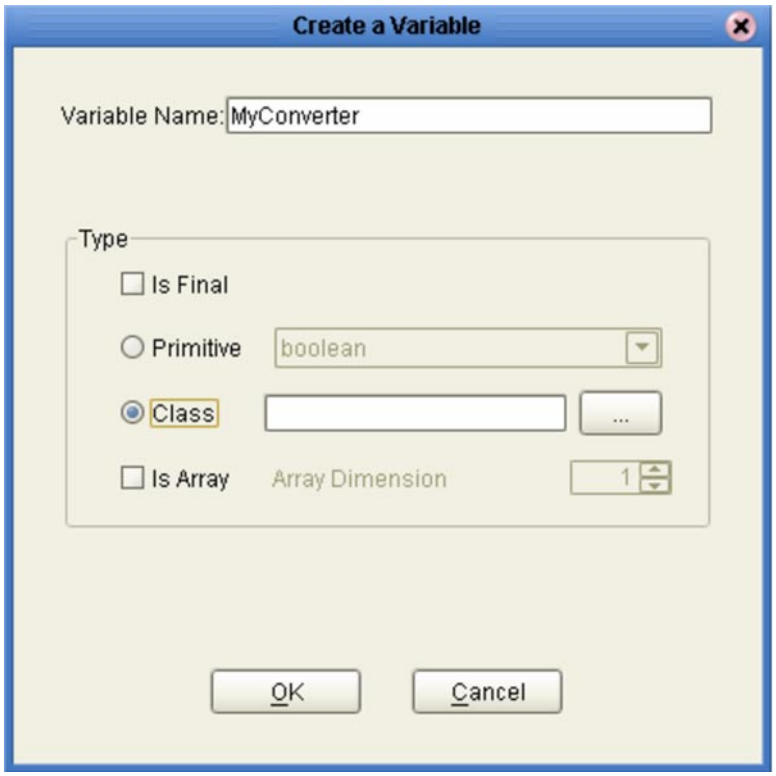
Figure 236 Using *unmarshalFromBytes* Method



2 Create a local variable of the Encoding Converter class.

A Click the **Local Variable** icon to display the *Create a Variable* dialog box (see Figure 237).

Figure 237 Create a Variable Dialog Box

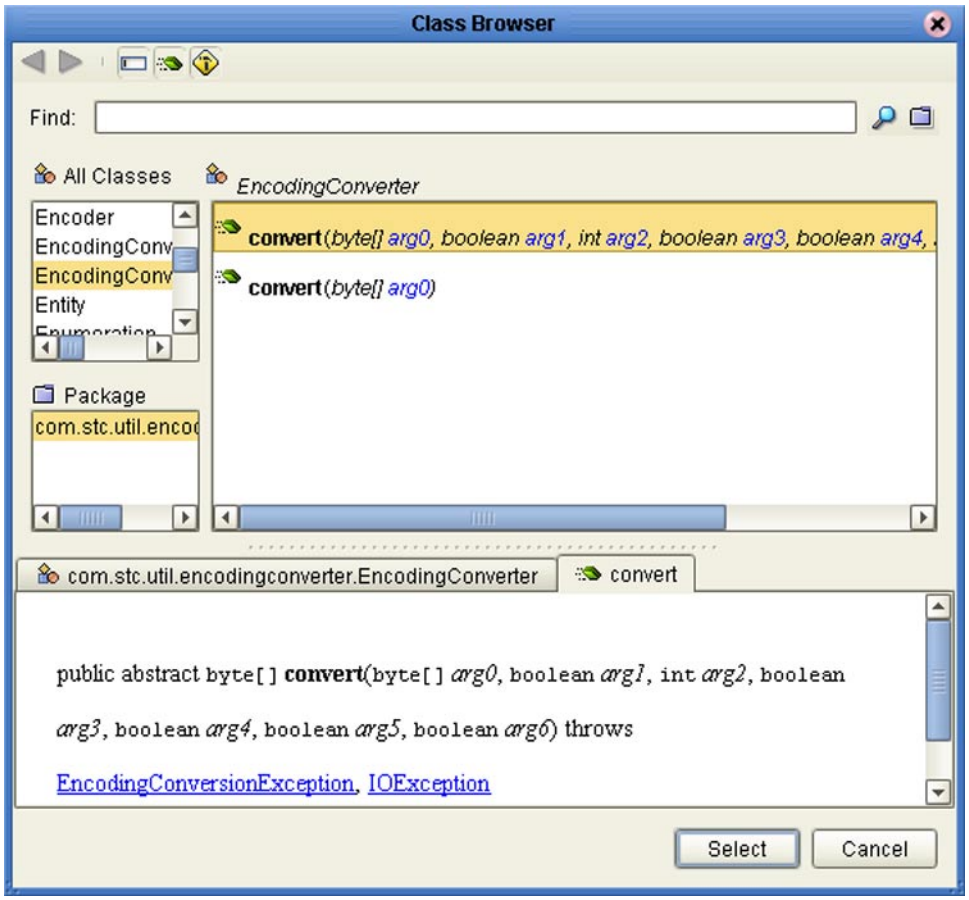


B Enter a name for the variable, for example: **MyConverter**.

C Click the **Class** option button to enable the entry field.

D Click the ellipsis (**…**) button to display the *Class Browser* dialog box (see Figure 238).

Figure 238 Class Browser Dialog Box

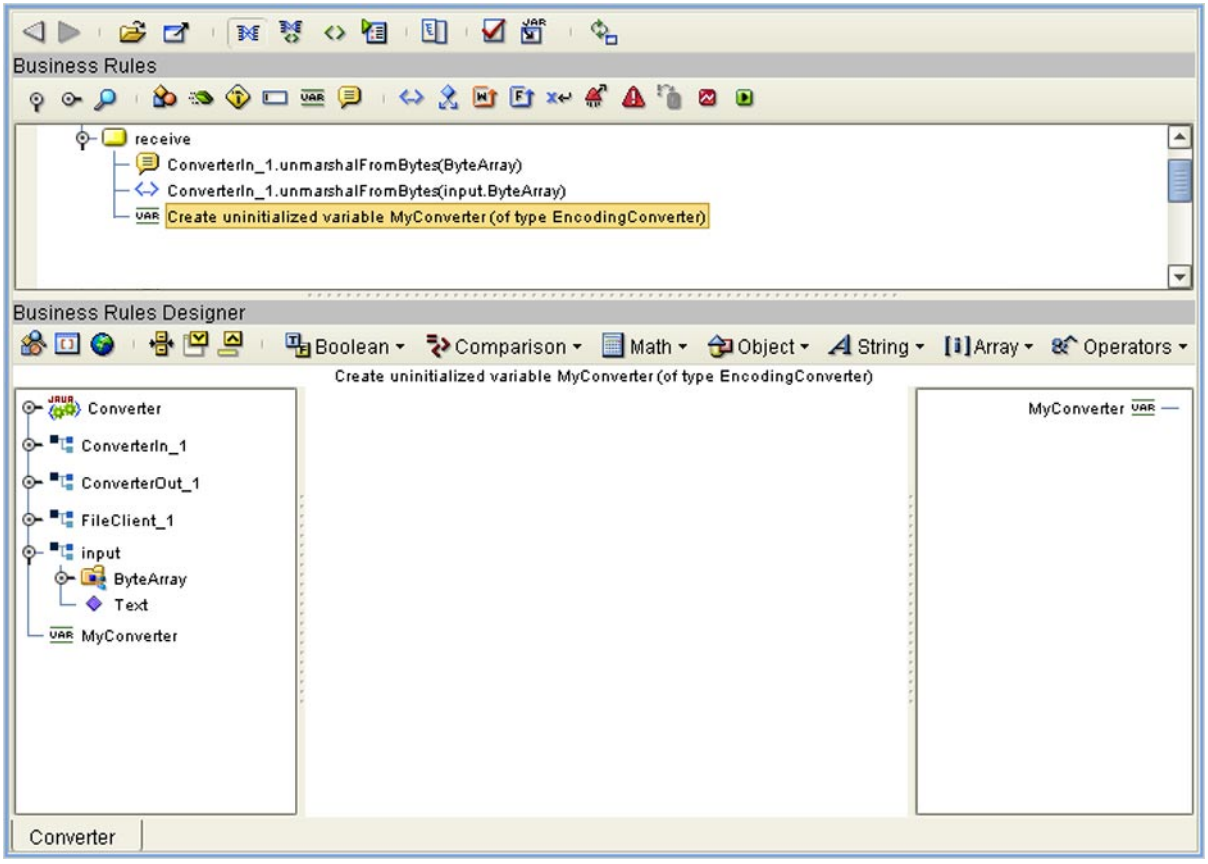


E Scroll to **Encoding Converter** and select it.

F Click **Select** to return to the **Create a Variable** dialog box.

G Click **OK** again to return to the main Collaboration Editor interface, where the local variable is now displayed (see Figure 239).

Figure 239 Local Variable *MyConverter* Added



3 Click the **Encoding Converter** icon (see Figure 240) to display the *Encoding Converter Method* dialog box shown in Figure 241.

Figure 240 Encoding Converter Icon

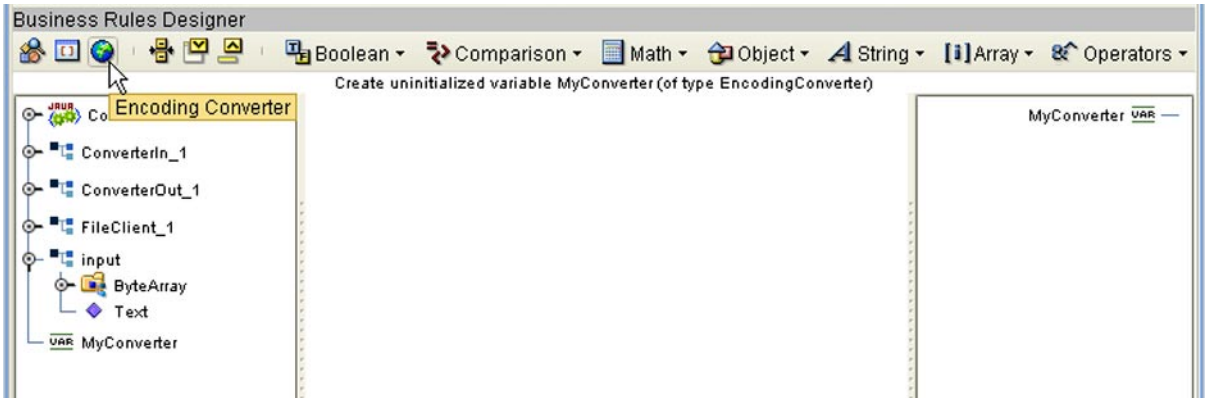
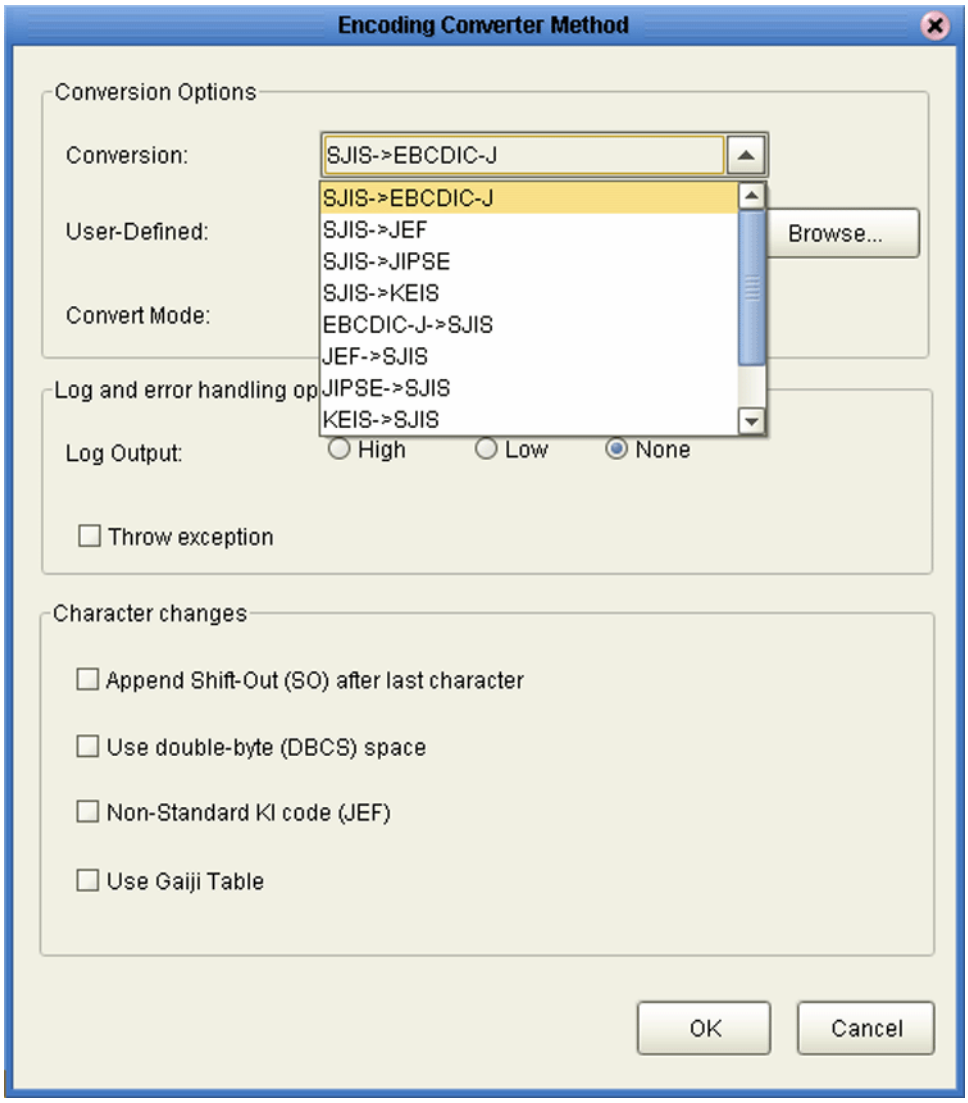


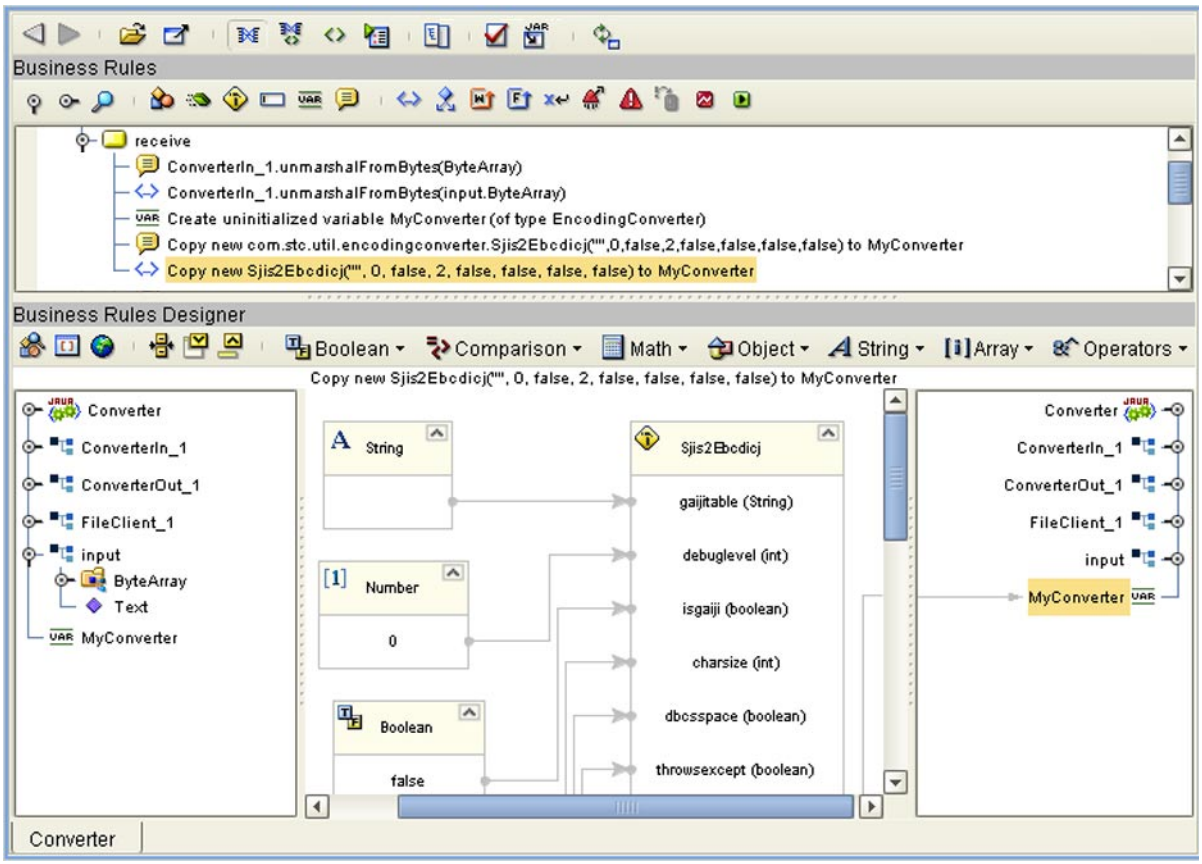
Figure 241 Encoding Converter Method Dialog Box



4 From the Conversion Options group box, select **SJIS->EBCDIC-J**, as shown in Figure 241, which creates the encoding converter rule shown in Figure 242.

5 Select any desired options from the other group boxes and click **OK**.

Figure 242 Encoding Converter Rule Created



6 Map the result to the local variable created in step 2 (**MyConverter**) in the right pane as shown in Figure 242.

7 Click on the variable **MyConverter** in the left pane to display its context menu and click **Select a method to call** (see Figure 243). This displays the dialog box shown in Figure 244.

Figure 243 Select a Method to Call

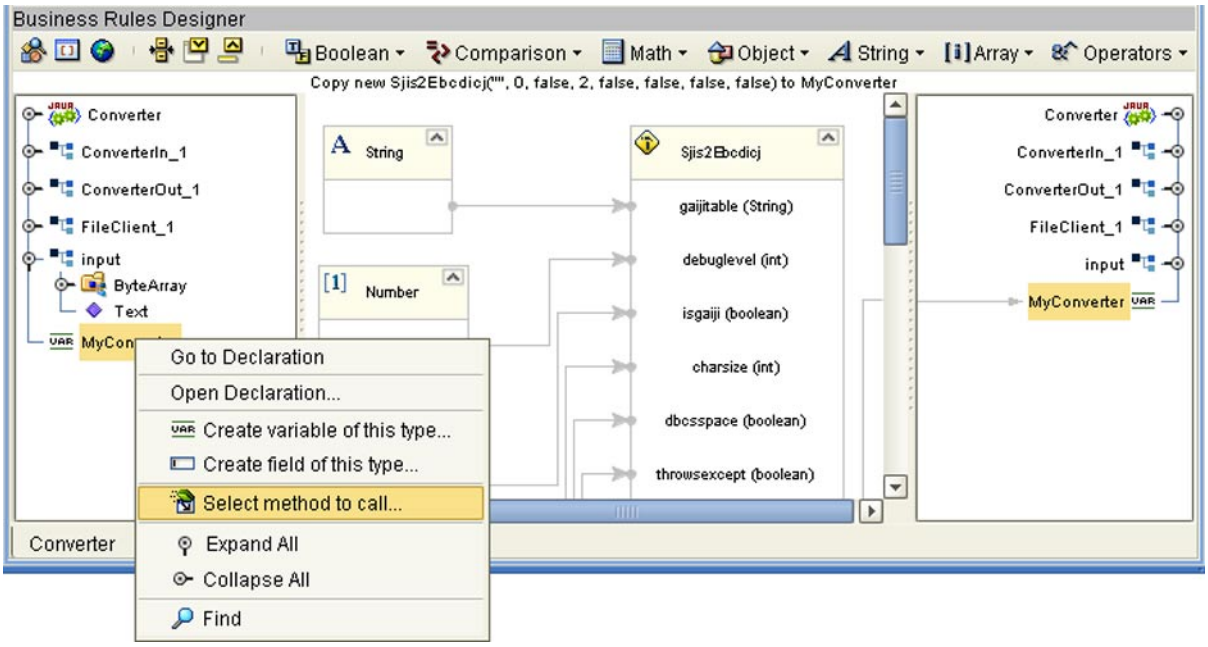
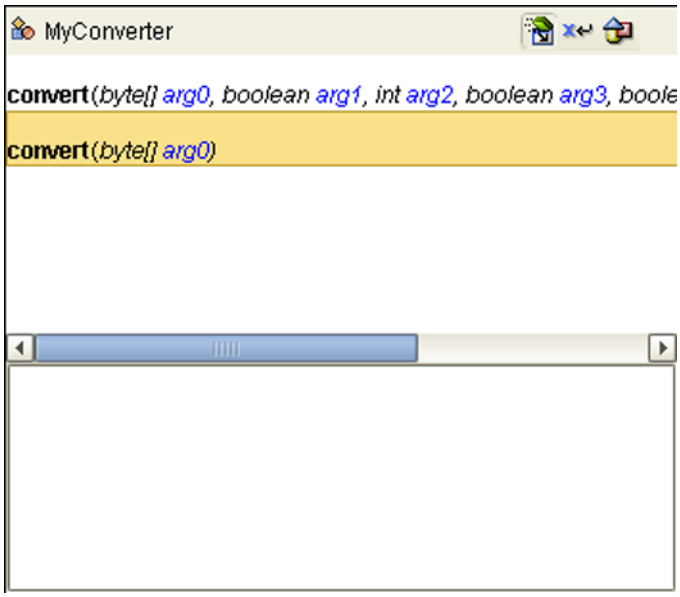


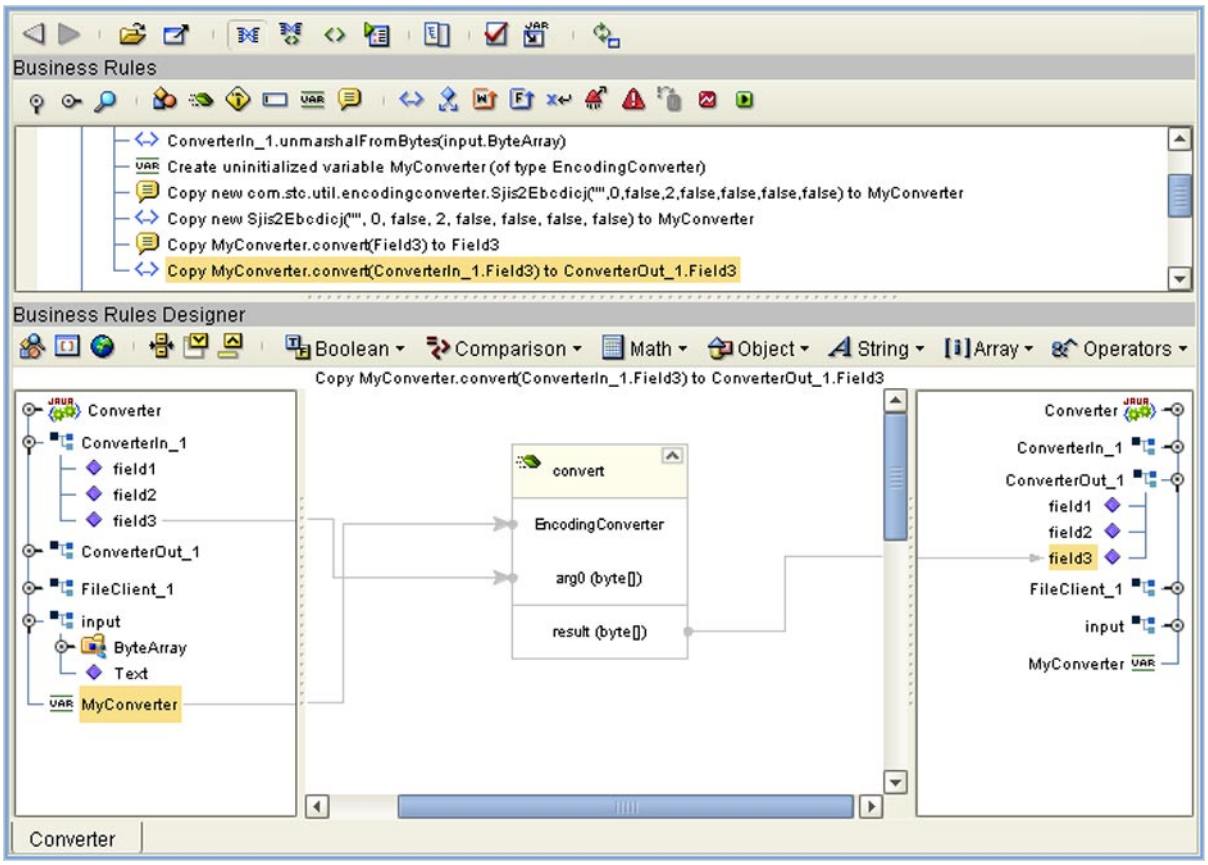
Figure 244 *Convert* Method List



8 Select **convert**(*byte[] arg0*), which places a *Convert* method box onto the mapping canvas.

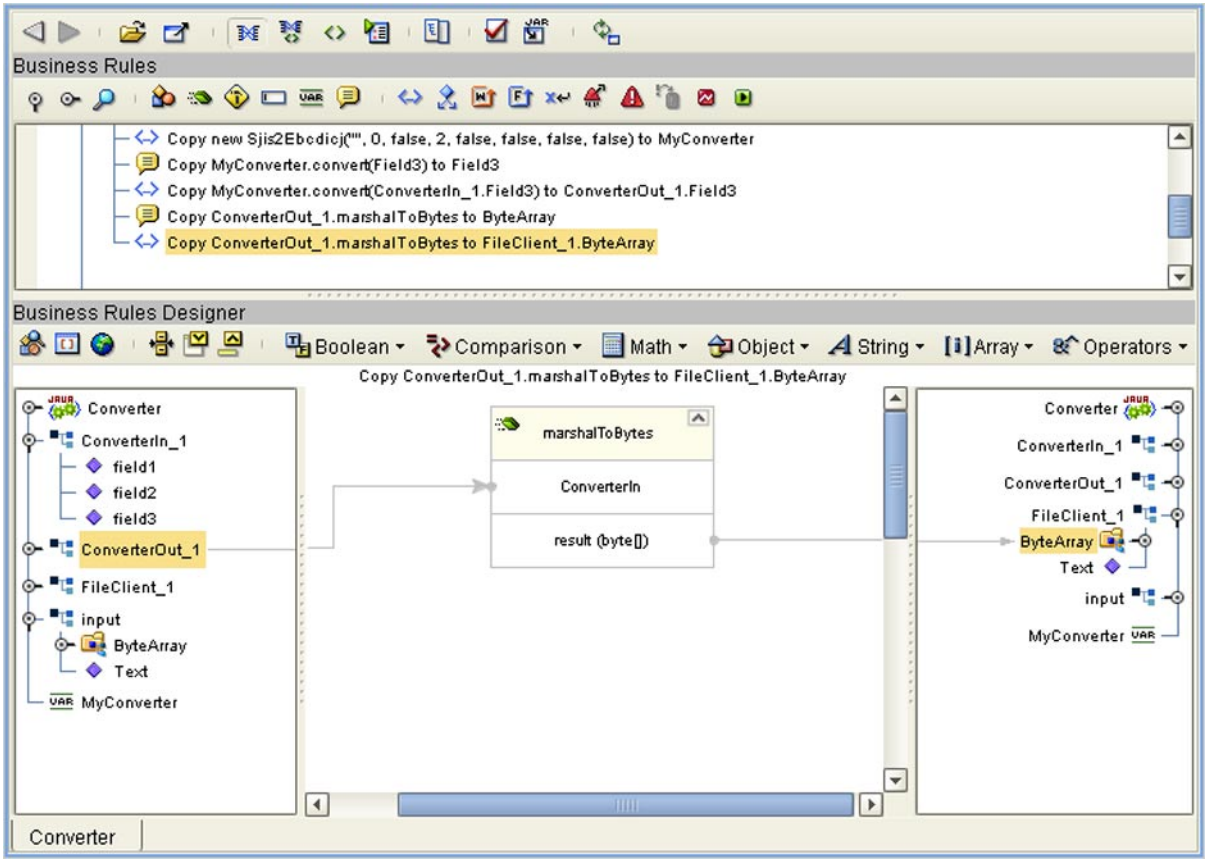
9 Map the node that contains data to be converted (**ConverterIn_1 / field3**) to **arg0** (see Figure 245), and map the **result** to the node that gets the converted data (**ConverterOut_1 / field3**).

Figure 245 Data Mapping



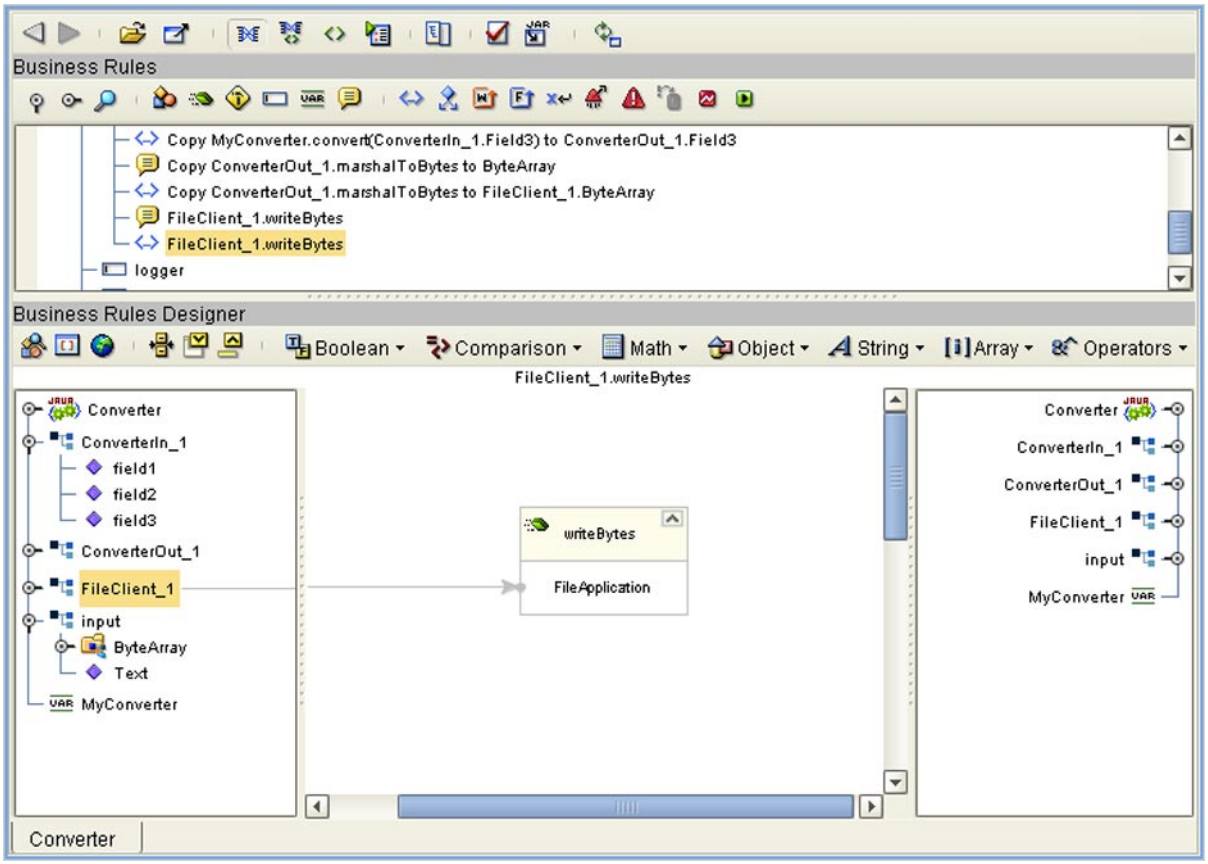
10 When finished with the Collaboration rules, marshal the output data using the **marshalToBytes** Java method as shown in Figure 246.

Figure 246 Marshal Output Data



11 Finally, output the data using the **writeBytes** Java method as shown in Figure 247.

Figure 247 Output Data

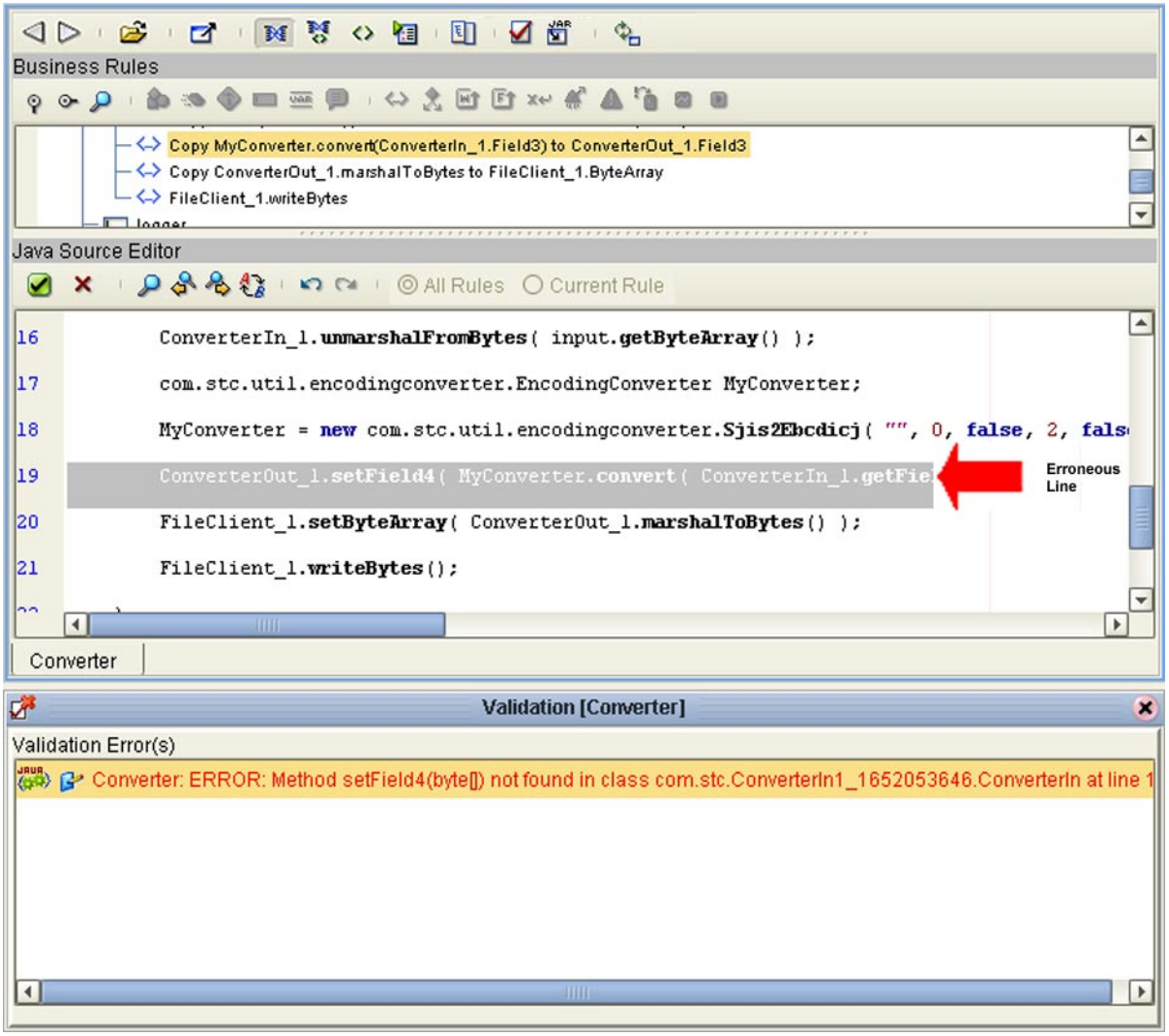


7.8.14 Validating Java-based Collaboration Definitions

Clicking the **Validate** icon in the Collaboration Editor Toolbar allows you to "precompile" the Java-based Collaboration Definition and display the errors in a validation panel below the Java Source Editor. To locate the error, double-click on the error message and the erroneous line of code will be identified as shown in Figure 249.

Figure 248 Validate Icon

Figure 249 Validating Java Code

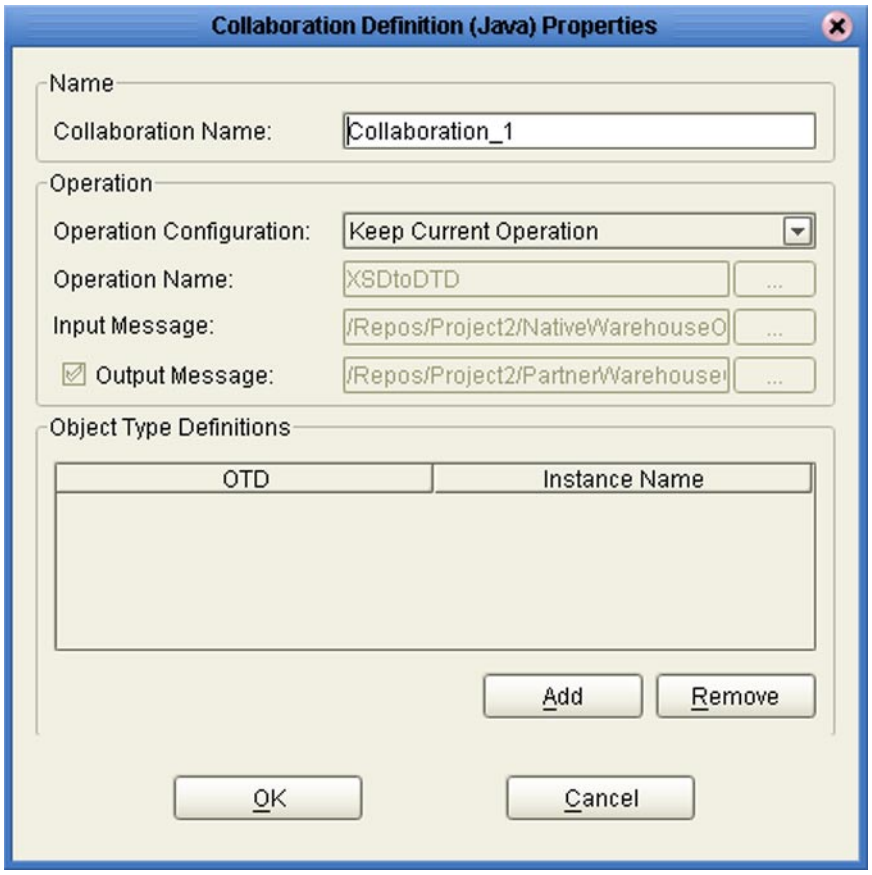


7.8.15 Editing Collaboration Definition Properties

Right-clicking a Collaboration Definition (Java) in the Project Explorer displays the context menu shown in Figure 59 on page 98. Selecting **Properties** from the menu displays the Collaboration Definition (Java) Properties dialog box for the selected Collaboration Definition (see Figure 250).

This dialog box resembles the Collaboration Definition Wizard, and shows the property values that were set previously. By default, the Operation Configuration field is set to **Keep Current Operation**; to edit the property values, you must select another command.

Figure 250 Collaboration Definition (Java) Properties



Important: *If you change any Java-based Collaboration that implements existing Web Services, be sure to reset the Web Service in the Collaboration Definition (Java) Properties dialog box, as described in the following procedure.*

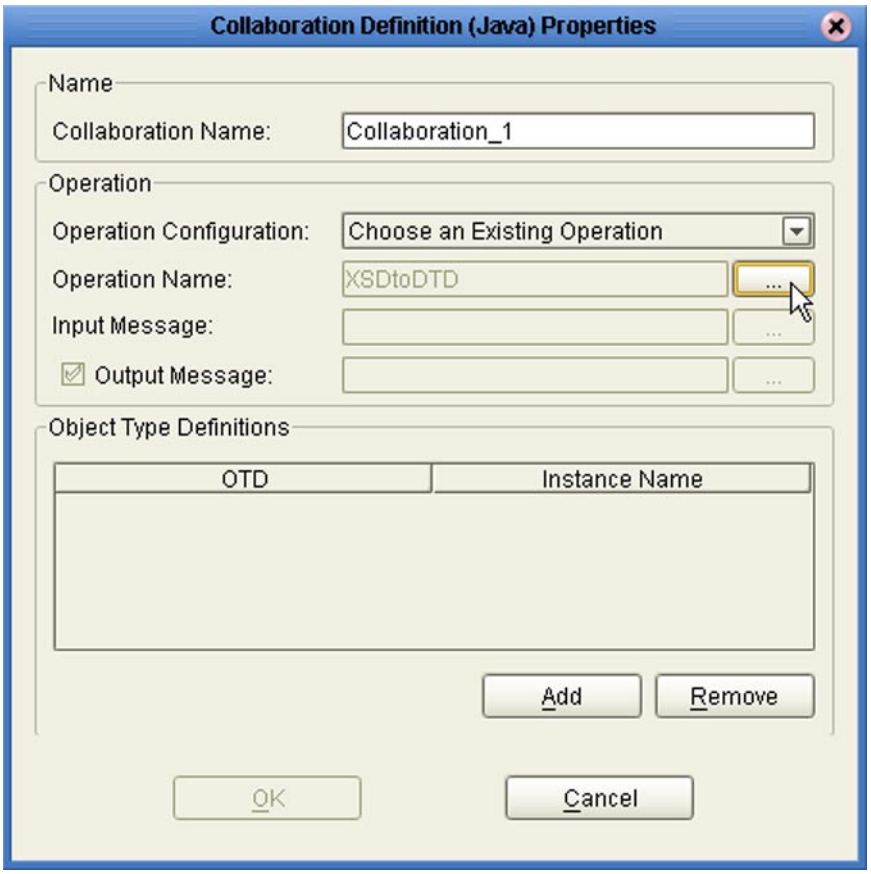
To reset the configuration properties

1 In the Explorer, right-click the Collaboration Definition that has been changed.

2 Select **Properties** from the Collaboration Definition context menu (see **Figure 59 on page 98**).

3 Select **Choose an Existing Operation** from the Operation Configuration list (see Figure 251).

Figure 251 Collaboration Definition (Java) Properties



4 Select the same Web Service Operation as before in **Operation Name** (click the ellipsis **[…]** button, as shown in Figure 251, to display a dialog box for selecting the operation).

5 Select the appropriate input and output messages.

6 Click **OK** to finish.

7.9 Using the Java Debugger

The Java Debugger enables you to debug Java-based Collaboration Definitions as deployed within an integration server on a Logical Host, in a run-time Environment. As such, it offers an alternative to creating logs or warnings in an Java-based Collaboration and subsequently inspecting them via the Enterprise Manager.

Although there are similarities between the Java Debugger and the Collaboration Tester (see **Using the Collaboration Tester (Java)** on page 242), they are separate tools for use in different situations. Both are displayed within the Collaboration Editor (Java).

7.9.1 Enabling the Debugger

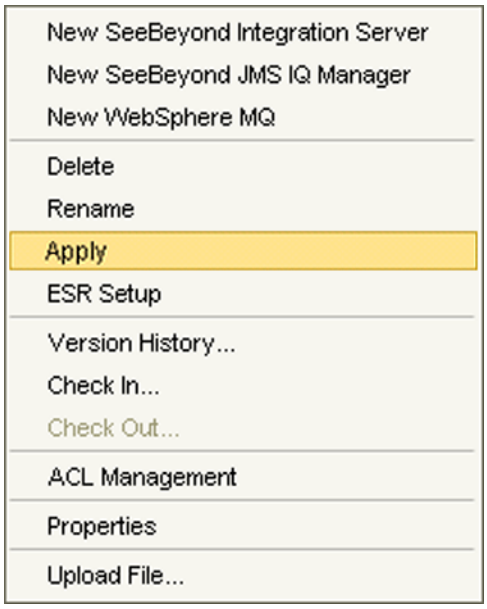
To enable the Java Debugger

1. In Enterprise Explorer, select the appropriate integration server.
2. Right-click to display the context menu.
3. Click **Properties** to display the Properties Dialog Box (see Figure 252).

Figure 252 Integration Server Properties Dialog Box

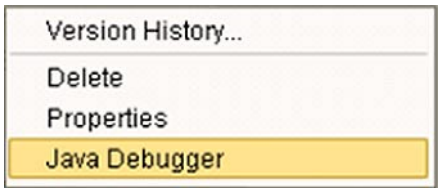
4. Set the value for *Attach Debugger* property to **True** and click **OK**.
5. In Enterprise Explorer, select the Logical Host containing the integration server you have just configured.
6. Right-click to display the context menu and click **Apply** (see Figure 253).

Figure 253 Logical Host Context Menu



7 The **Java Debugger** option now is enabled in the integration server context menu (see Figure 254).

Figure 254 Integration Server Context Menu

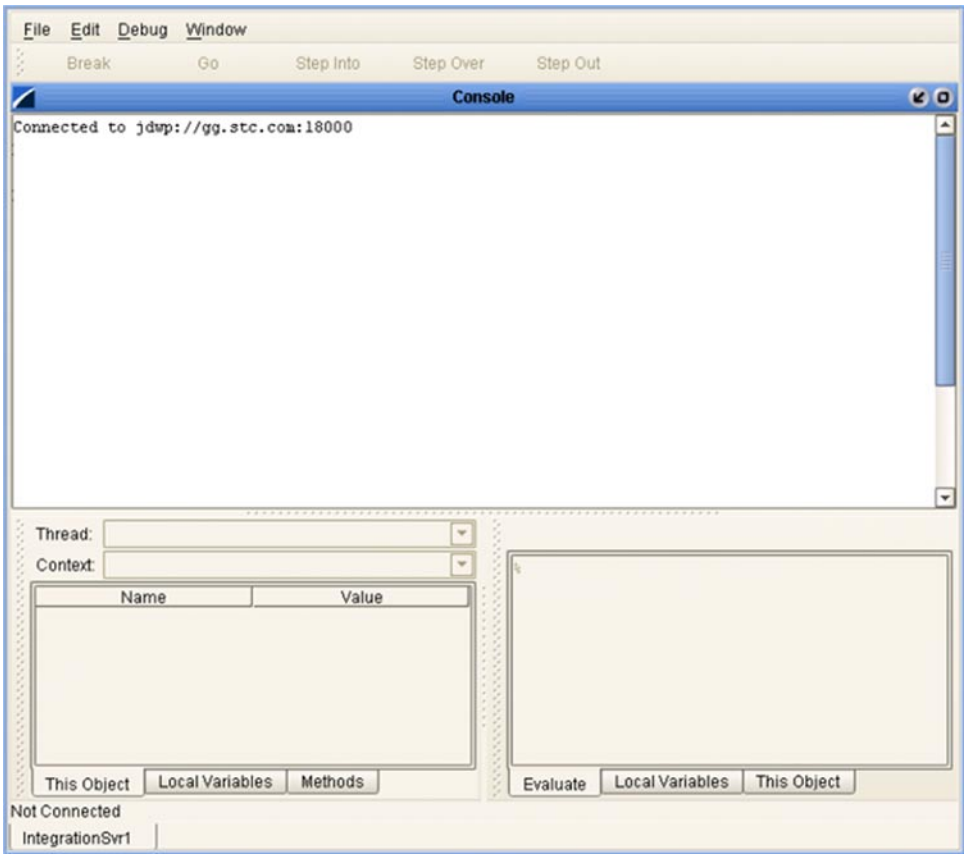


7.9.2 Invoking the Java Debugger

To invoke the Java Debugger

1. In Enterprise Explorer, select the appropriate integration server.
2. Right-click to display the context menu.
3. Click **Java Debugger** (see Figure 254), and the Java Debugger appears in the Enterprise Designer Editor panel (see Figure 255).

Figure 255 Java Debugger



Note: *The Java Debugger appears whether or not the connection was successful. If there is no* ***Connected to…*** *message, try the following procedure:*

A. Select **Attach to JVM…** from the File menu (see Figure 256), which displays the *Attach to JVM* dialog box (see Figure 257).

Figure 256 File Menu

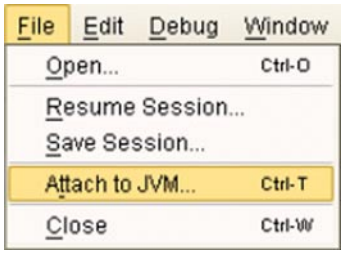
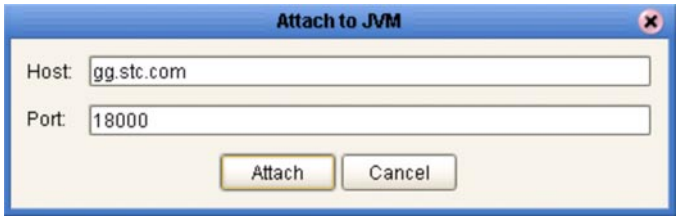


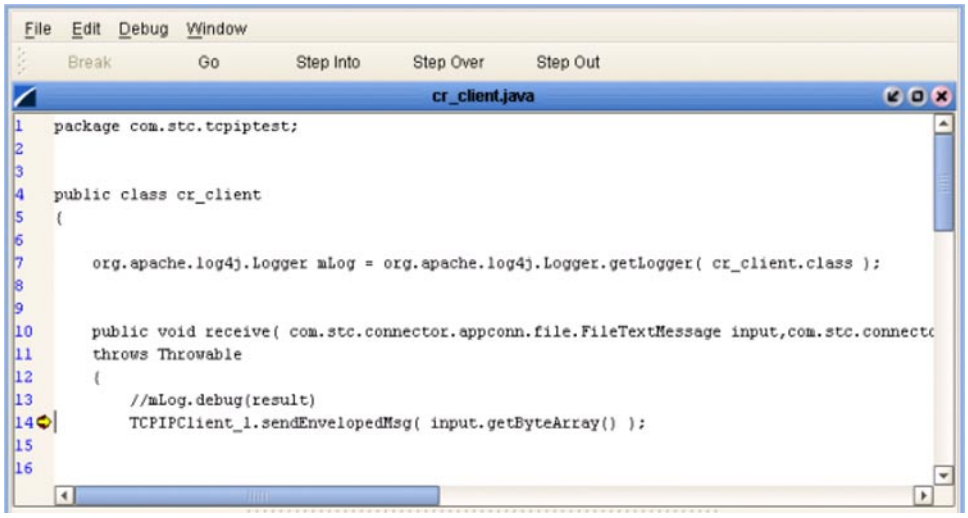
Figure 257 Attach to JVM Dialog Box



B Enter the integration server's host name and port number into the text boxes and click **Attach**. The debugger then re-attempts to connect to the integration server.

4 Once the Java Debugger is running, Java source code is displayed as soon as a Java-based Collaboration executes (see Figure 258).

Figure 258 Java Source Code Display



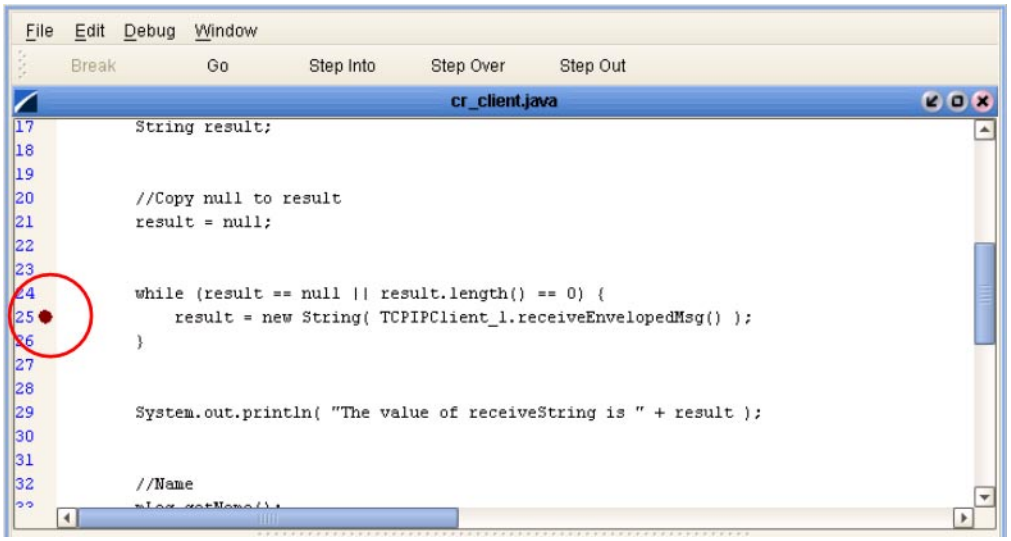
5 You can now set breakpoints to assist in examining and debugging the code.

7.9.3 Setting Breakpoints

To set a breakpoint

1 Click next to a line number in the executed source code. A red dot is displayed as a marker (see Figure 259).

Figure 259 Breakpoint Example



2 Alternatively, you can set stops in a specific class or method, or have the debugger break on an exception. These options are available from the **Debug** menu.

A To set a stop in a method, for example, select **Stop in Method…** from the Debug menu (see Figure 260). A *Stop in Method* dialog box is displayed, in which you can select the desired method (seeFigure 261).

Figure 260 Debug Menu

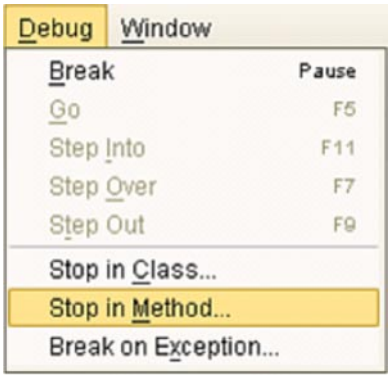
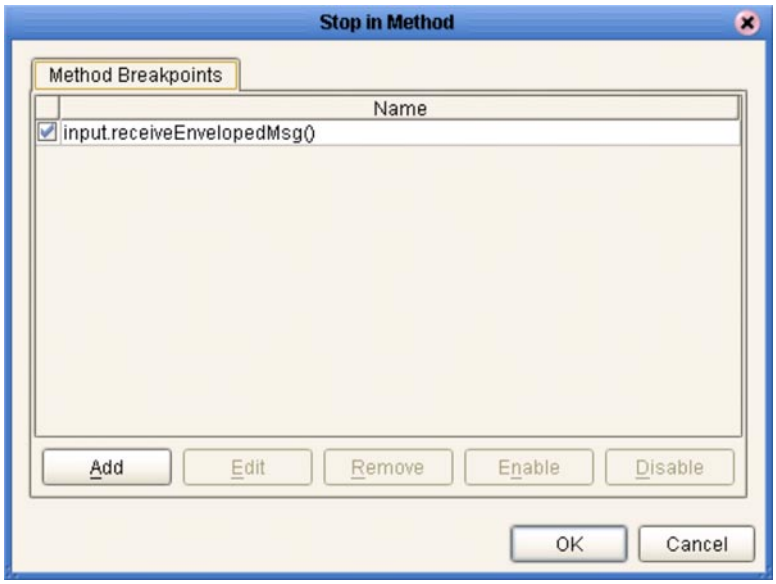


Figure 261 Stop in Method Dialog Box



B To break on an exception, select **Break on Exception…** from the Debug menu, which displays a *Choose Exception* dialog box (see Figure 262). All occurrences of the specified exception are then trapped and reported (see Figure 263).

Figure 262 Choose Exception Dialog Box

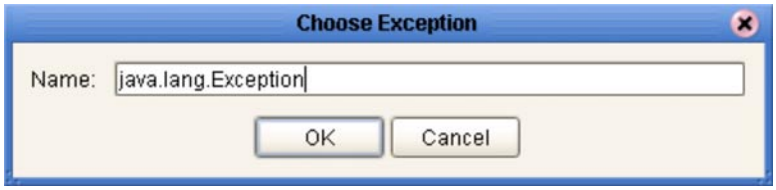
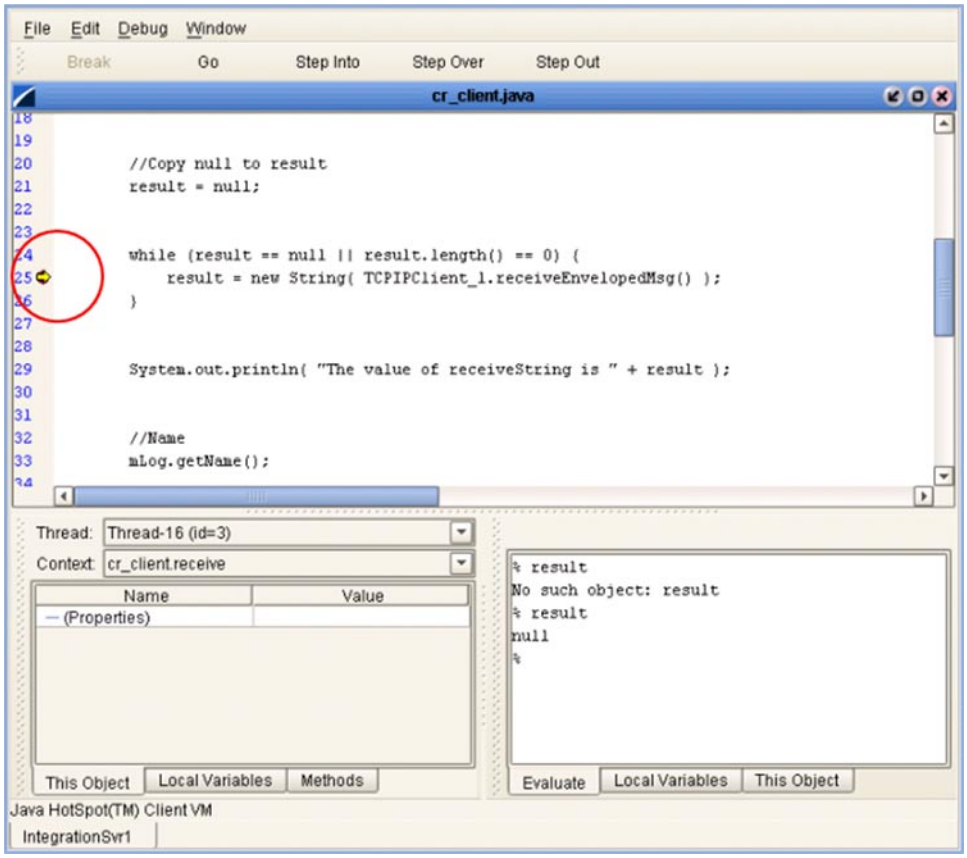


Figure 263 Break on Exception Dialog Box

7.9.4 Inspecting and Editing the Source Code

As soon as the execution of the Java-based Collaboration arrives at a set breakpoint, it stops executing and displays an right arrow indicator next to the line number in the source code (see Figure 264).

Figure 264 Breakpoint Indicator

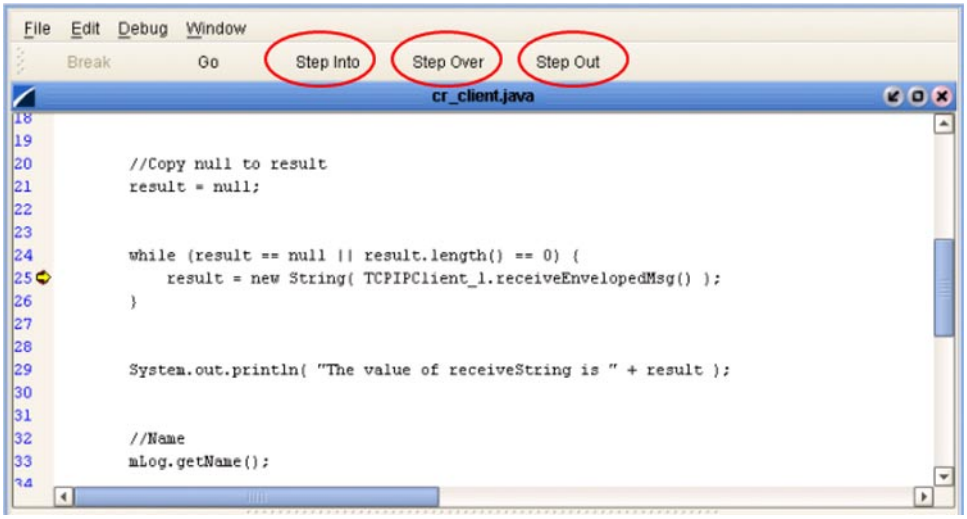


At this point, you can continue by (for example):

- Stepping Into
- Stepping Over
- Stepping Out
- Inspecting a local variable
- Setting a local variable

Stepping Into, Over, or Out

Figure 265 Stepping Into, Over, and Out Commands

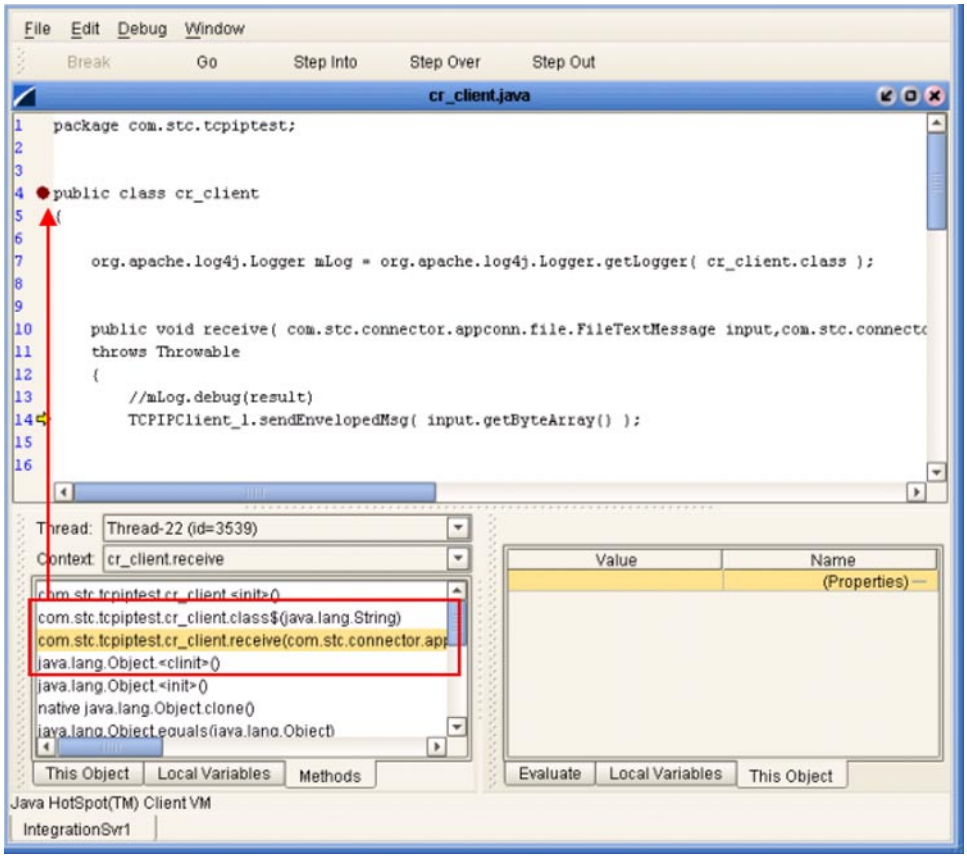


- By selecting the **Step Into** option (see Figure 265), the breakpoint is lifted and execution of the Collaboration will continue, *including* the line of code at the breakpoint.
- By selecting the **Step Over** option, the breakpoint is lifted and execution of the Collaboration will continue, *ignoring* the line of code at the breakpoint.
- By selecting the **Step Out** option, execution of the Collaboration is terminated.

Inspecting Java Threads and Methods

Selecting the **Methods** tab in the left bottom panel of the debugger displays the currently executed Java thread and method (see Figure 266).

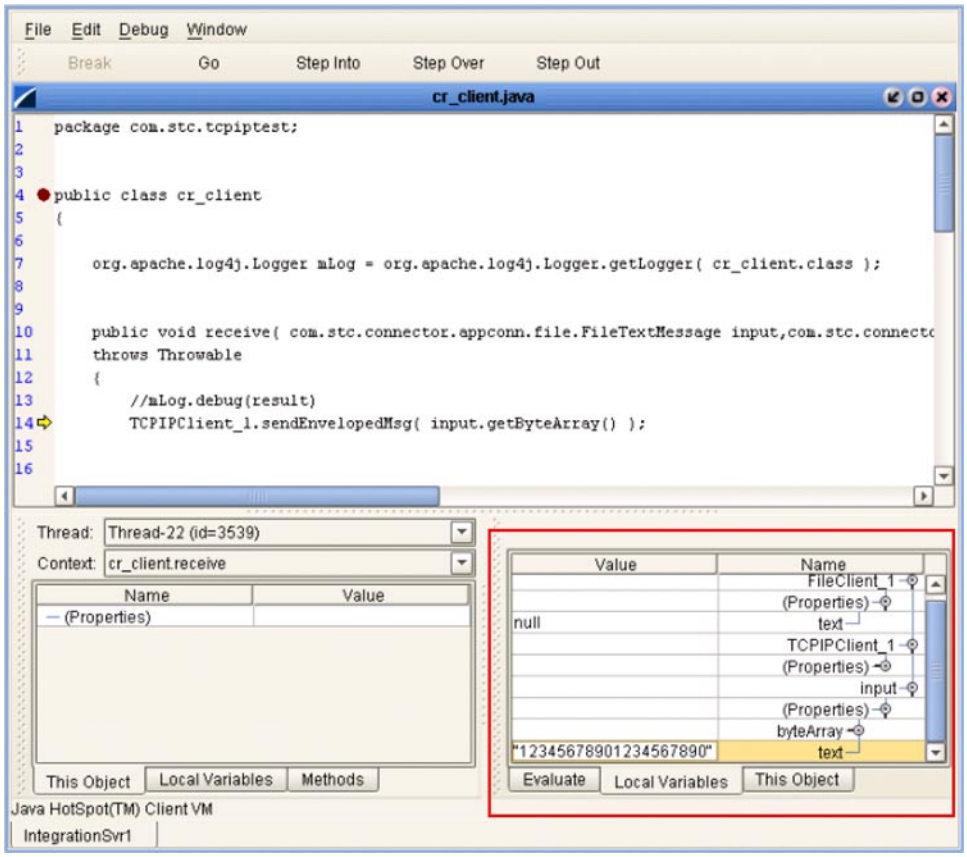
Figure 266 Java Thread and Method Display



Inspecting a Local Variable or Method

You can inspect a local variable by selecting the **Local Variables** tab in the right bottom panel of the debugger (see Figure 267). All nodes of the currently executed Java class are displayed here, and you can expand or collapse certain nodes to search for the value of the desired variable.

Figure 267 Local Variables Tab

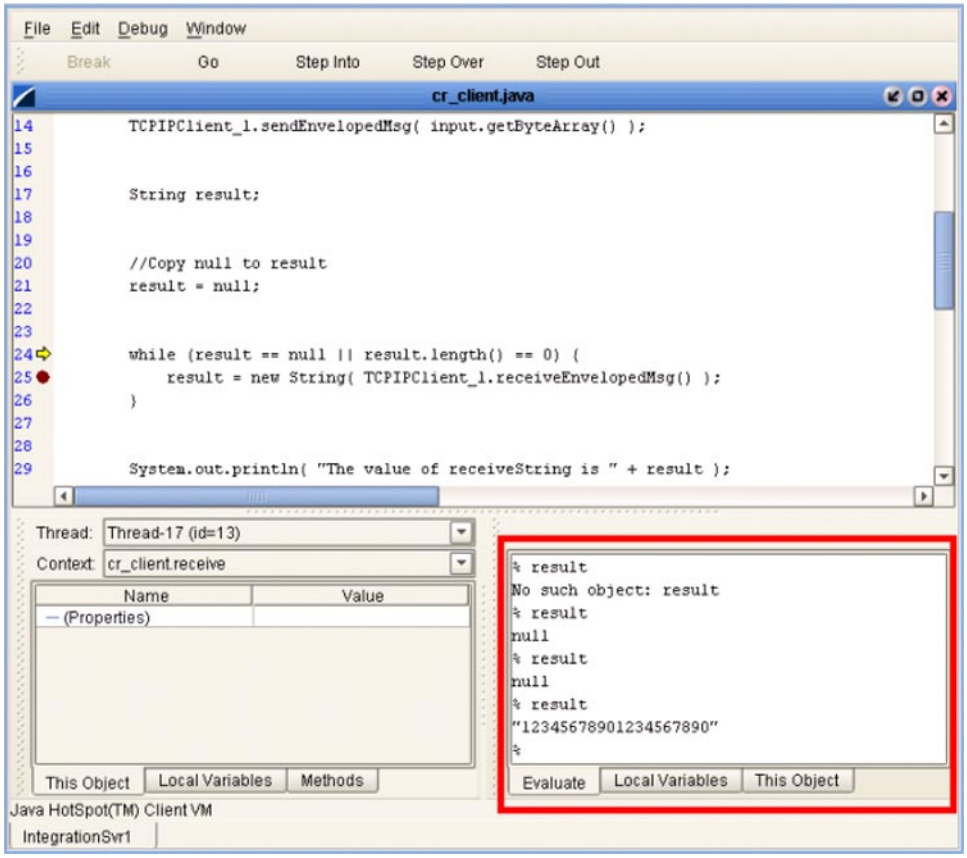


You also can inspect a local variable by selecting the **Evaluate** tab and entering the variable name in the panel using the following syntax:

```
% <variable_name>
```

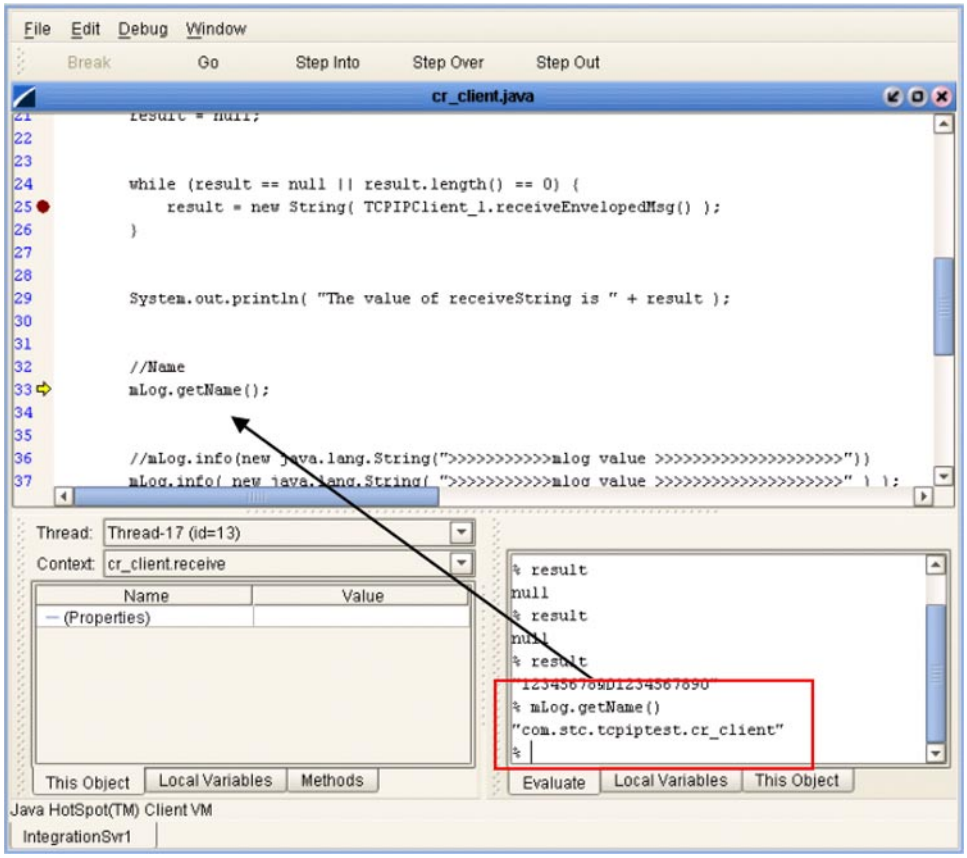
If the code has not initialized a variable, *No such object* is displayed; otherwise, the current value of the variable is displayed (see Figure 268).

Figure 268 Evaluate Local Variable



You can also inspect the result of a method by entering the method name, following the same syntax format (see Figure 269).

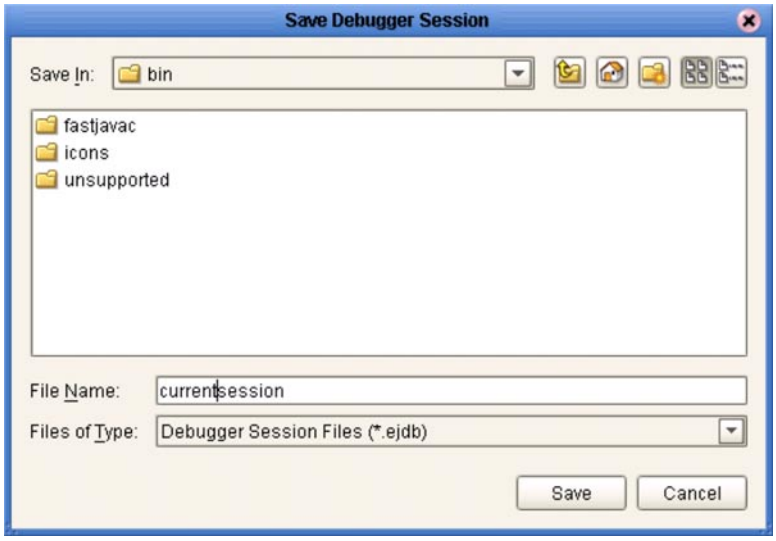
Figure 269 Evaluate Method



Saving and Resuming Debug Sessions

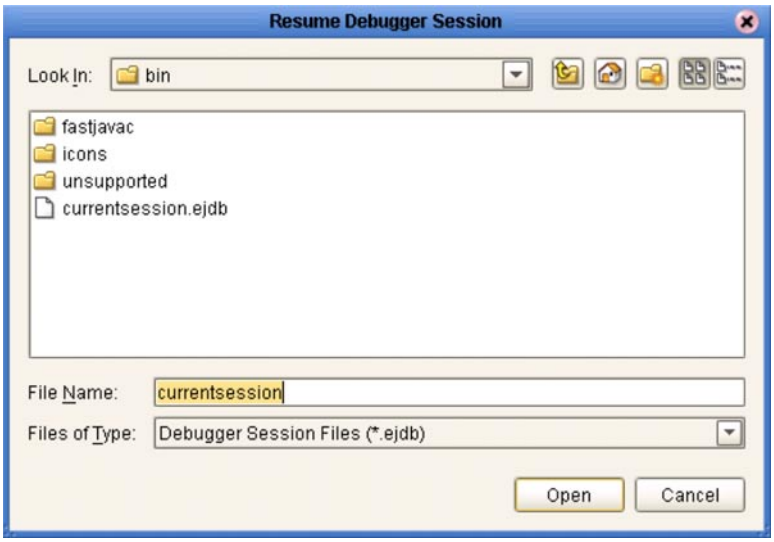
You can pause the debugging process by saving the session to a file. Selecting **Save Session** from the debugger File menu displays the *Save Debugger Session* dialog box (see Figure 270), in which you specify the file name and location.

Figure 270 Save Debugger Session Dialog Box



You can continue a debugging process that was paused by saving to a file. Selecting **Resume Session** from the debugger File menu displays the *Resume Debugger Session* dialog box (see Figure 271), in which you specify the file name and location.

Figure 271 Resume Debugger Session Dialog Box



Chapter 8

Collaboration Definitions (XSLT)

This chapter describes the features of the Collaboration Definition Editor (XSLT) and outlines procedures for building XSLT-based Collaboration Definitions.

What's in This Chapter

8.1 Overview

Collaboration Definitions define how data should be processed and routed between Project components, how databases should be queried in response to requests, and how APIs to one or more applications should be invoked. The external data formats that characterize the input and output data structures in a Collaboration Definition are described by Object Type Definitions (OTDs).

A Collaboration will typically receive a message containing the external representation of a particular OTD. It will use the *unmarshal* method of an instance of that OTD to parse the data and make it accessible though the hierarchical data structure. Then it will perform some operation—for example, copying parts of the data to another OTD instance. Finally, it will invoke the *marshal* method on the other OTD instance to render the contents of its data structure as a single, serialized data stream for further transport.

At run time, an OTD instance is accessed from BPEL using XPath expressions. Each of the nodes comprising the hierarchy of the data structure has a set of properties with *get* and *set* methods. See **Object Type Definitions** on page 119 for additional information.

Important: *If you delete an OTD in the Project Explorer, any Collaboration Definitions that have been built using that OTD will be affected. It is recommended that you run the Impact Analyzer before attempting to delete any OTDs (see* **Impact Analyzer** *on page 77).*

As with other eGate Integrator components, it is essential to manage versions of Collaboration Definitions carefully. See **Version Control** on page 79 for descriptions of various version control features applicable to Collaboration Definitions.

The Enterprise Designer includes two primary tools, the Collaboration Definition Wizard (XSLT) and Collaboration Editor (XSLT), that are used to create and customize your XSLT-based Collaboration Definitions. These tools are described in the following sections.

8.2 Using the Collaboration Definition Wizard (XSLT)

The Collaboration Definition Wizard (XSLT) guides you through the initial phases of creating an XSLT-based Collaboration Definition, and then invokes the Collaboration Editor (XSLT). The user interface is highly self-explanatory, but details of the navigation buttons are listed in Table 80 for your reference.

Table 80 Wizard Navigation Buttons

Button	Function
< Back	Returns to the previous step in the wizard. This button is disabled on the first step.
Next >	Goes to the next step in the wizard. This button is disabled on the last step.
Finish	Saves all Collaboration Definition settings and closes the wizard. This button is only enabled on the last step.
Cancel	Closes the wizard without saving the Collaboration Definition.
Help	Displays the online help documentation for the Collaboration Definition Wizard dialog box.

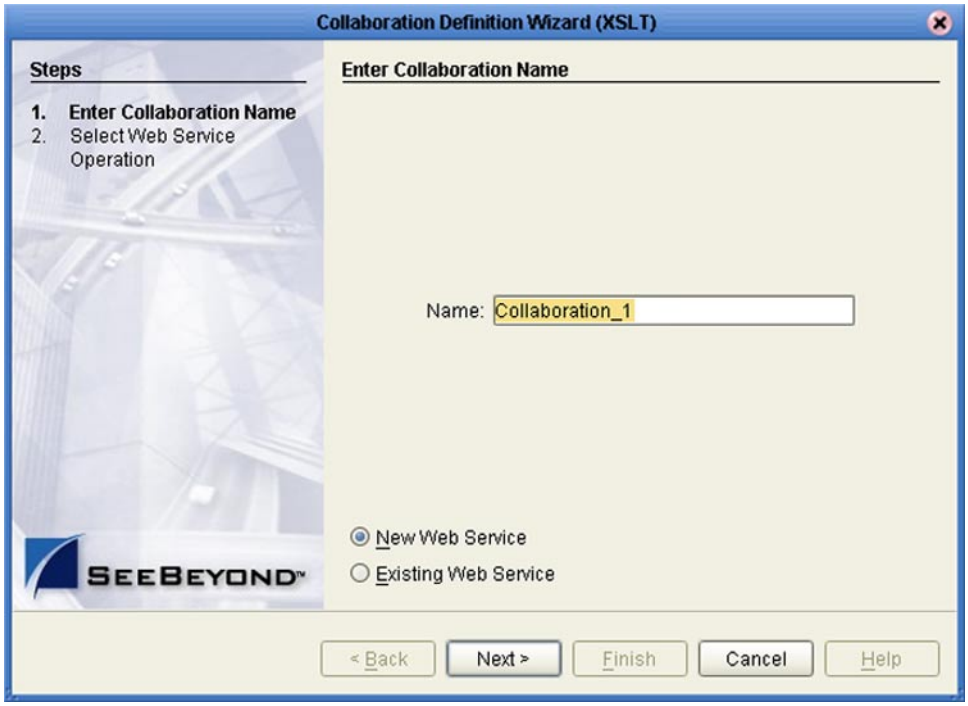
8.2.1 Creating a Collaboration Definition (XSLT)

Note: *Any changes made to the names of the Collaborations should be done when the Collaboration is created. If the name is changed later, the Collaboration should be opened, regenerated if applicable, and saved again. This procedure should also be performed before creating the Connectivity Map and Deployment Profile.*

To create an XSLT-based Collaboration Definition

1. Right-click on a Project in the Enterprise Explorer to display the Project context menu.
2. Select **New > Collaboration Definition (XSLT)** to invoke the Collaboration Definition Wizard (XSLT).
3. Enter a **Name** for your Collaboration, as shown in Figure 272.

Figure 272 Collaboration Wizard (XSLT) Dialog Box



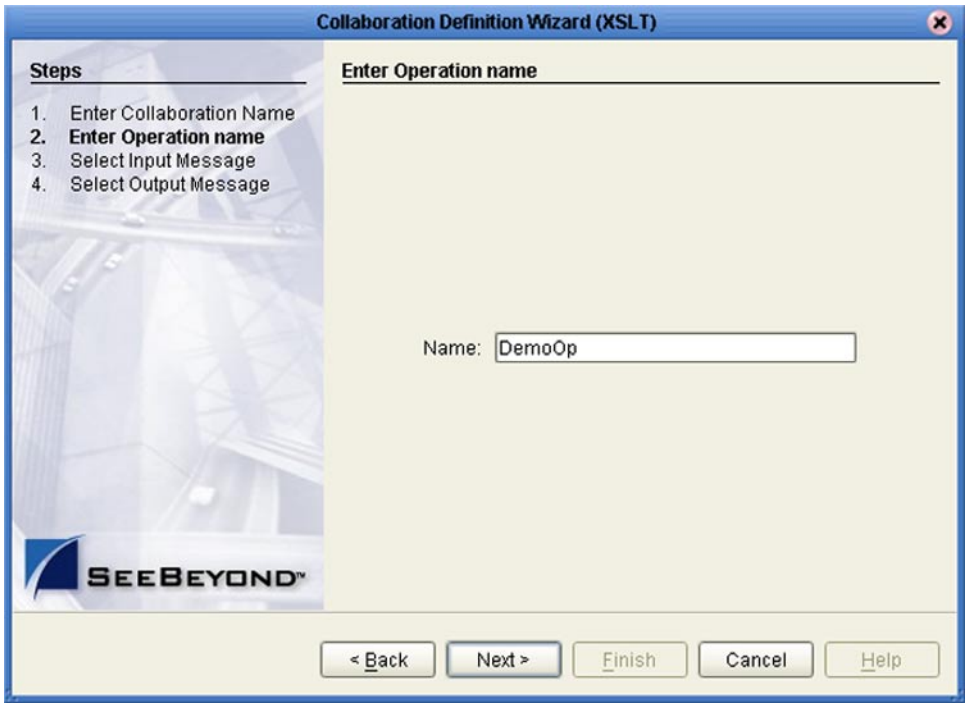
4. Select a Web service, which can be either:
   - A New Web Service.
   - An Existing Web Service (for example, an eInsight process or a Java Web Service Operation).
5. Click **Next** to proceed to the next Wizard dialog, which is dependent upon your Web Service selection.

New Web Service

If you selected a New Web Service, you will be presented with the following set of Wizard dialogs.

1 Enter an operation name, as shown in Figure 273. This will become the *method* that can be used to invoke the XSLT-based Collaboration as a Web service.

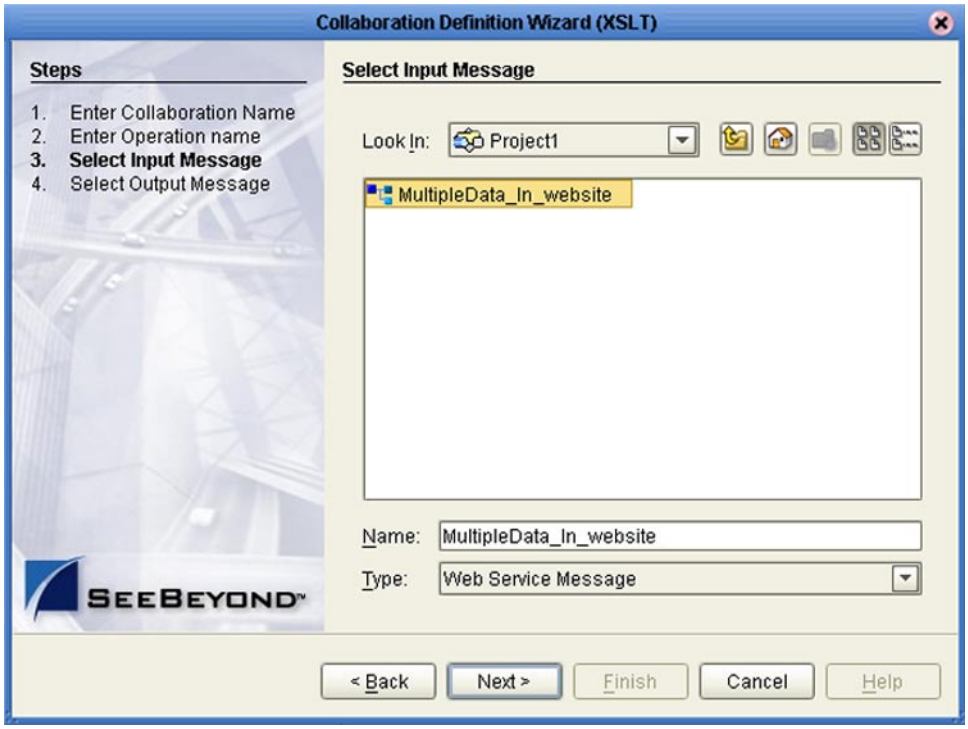
Figure 273 New Web Service: Operation Name



2 Click **Next** to proceed to the next Wizard dialog.

3 Select the input Web service message, as shown in Figure 274.

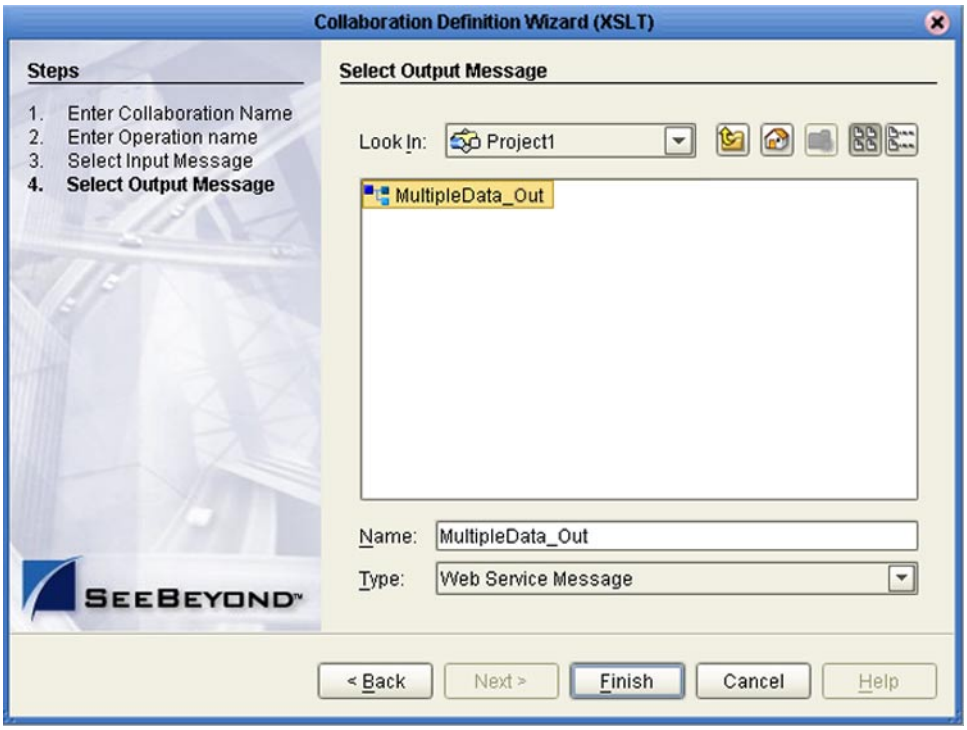
Figure 274 New Web Service: Input Message



4 Click **Next** to proceed to the next Wizard dialog.

5 Select the output Web service message, as shown in Figure 275.

Figure 275 New Web Service: Output Message



6 Click **Finish** to proceed to the Collaboration Editor (XSLT).

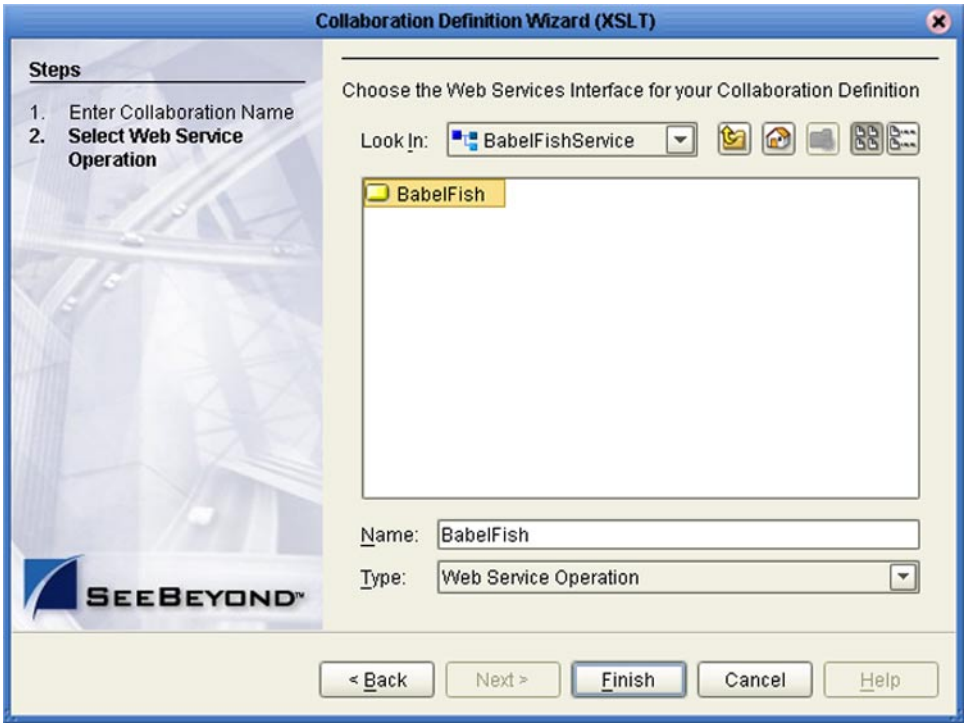
Existing Web Service

If you selected an Existing Web Service, you will be presented with the Wizard dialog shown in Figure 276.

1 Select a Web service operation, which can be either:

- An installed ICAN Web Service.
- A custom Web Service (for example, something that has been created in an ICAN Project).

Figure 276 Existing Web Service: Select Operation

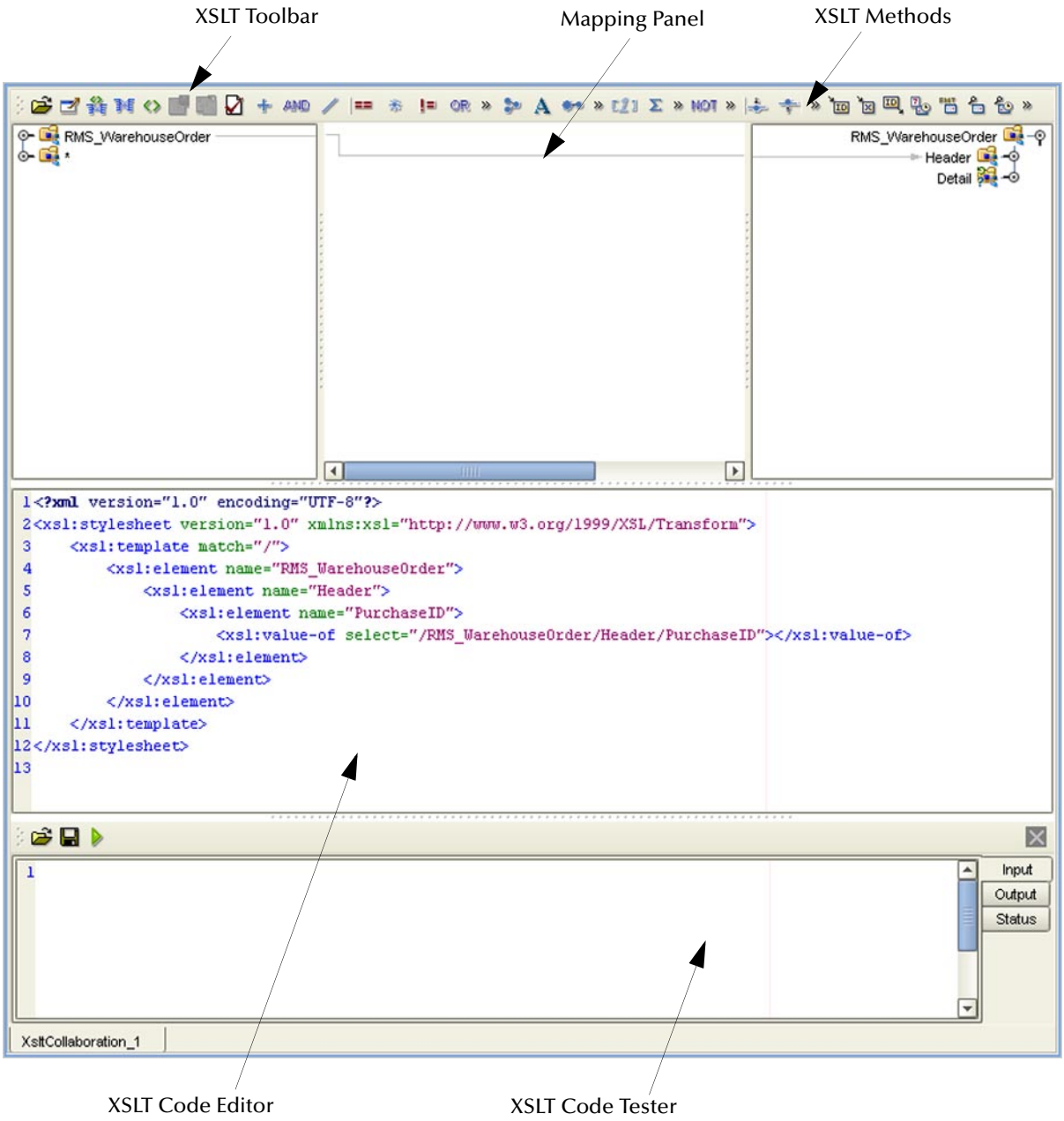


2 Click **Finish** to proceed to the Collaboration Editor (XSLT).

8.3 Using the Collaboration Editor (XSLT)

After you have created an XSLT-based Collaboration Definition using the Collaboration Definition Wizard (XSLT), the Collaboration Editor (XSLT) appears in the Editor panel of the Enterprise Designer. Major features of this window are identified in Figure 277.

Figure 277 Collaboration Editor (XSLT)

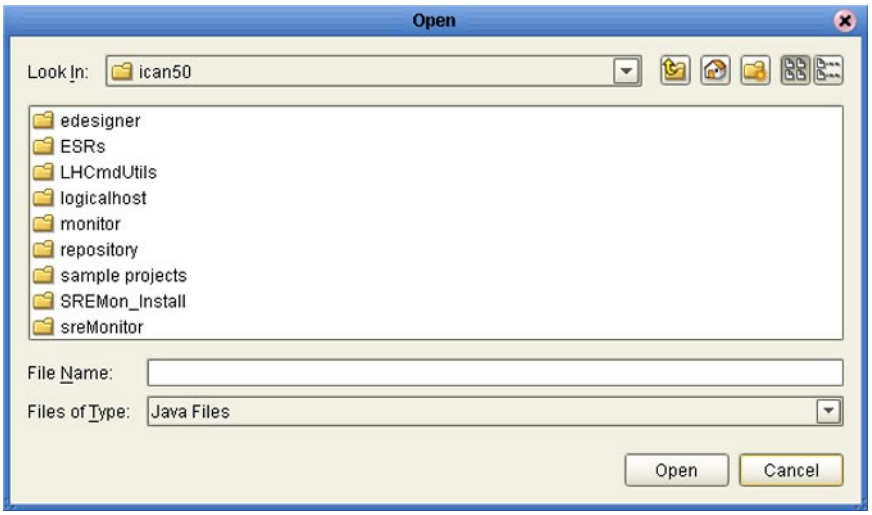


You can also invoke the Collaboration Editor (XSLT) by selecting **Open** in the context menu for an existing XSLT-based Collaboration Definition in the Enterprise Explorer.

The XSLT Mapping panel is used to map fields and add methods to the Collaboration Definition. At the top left of the Mapping panel is the toolbar, containing icons as described in Table 81. At the top right of the Mapping area is the XSLT Method Palette, which contains a collection of XSLT methods. The XSLT Code Editor panel allows you to view, enter and edit the XSLT code for the Collaboration Definition. The Tester panel allows you to run the XSLT code without deploying the Project.

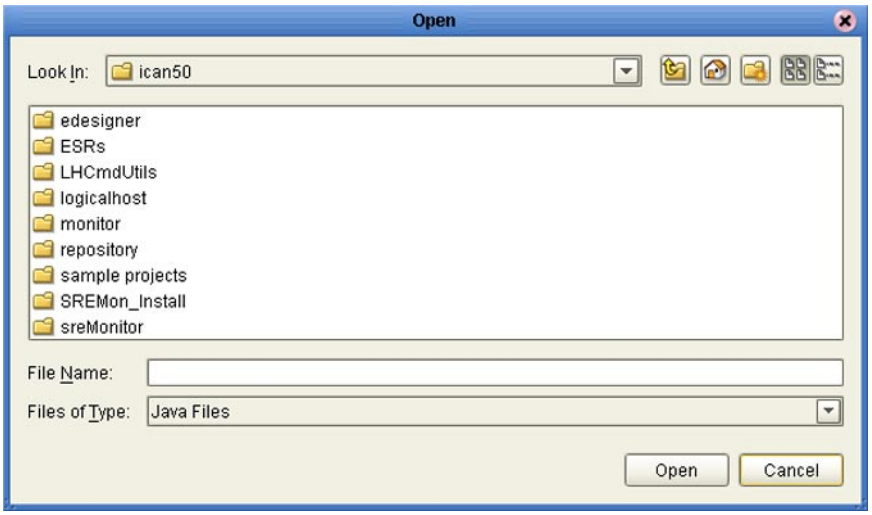
8.3.1 XSLT Toolbar Icons

Table 81 XSLT Toolbar Icons

Icon	Command	Function
	Import File	Displays the **Open** dialog box, which you can use to locate and select a Collaboration Definition (XSLT) to import. When you import a file, any previously generated code or rules are deleted. The imported code does not get appended to the existing Collaboration Rules. 
	Save XSLT to a Local File	Displays the **Save** dialog box, which you can use to save the selected XSLT-based Collaboration Definition to a file.
	Show Maps and Code	Displays both the Mapping and XSLT Code areas. This is the default view setting.
	Show Mapping Only	Displays the Mapping area and hides the XSLT Code area.
	Show XSLT Code Only	Displays the XSLT Code area and hides the Mapping area.
	Commit Code Changes	Commits changes made to the XSLT code since the last time it was committed. Changes will now be shown in the Mapping panel.
	Roll Back Code Changes	Cancels changes made to the XSLT code since the last time it was committed.
	Test XSLT Code	Displays the XSLT Tester panel.

8.3.2 XSLT Tester Icons

Table 82 XSLT Tester Icons

Icon	Command	Function
	Import File	Displays the **Open** dialog box, which you can use to locate and select a Collaboration Definition (XSLT) to import. When you import a file, any previously generated code or rules are deleted. The imported code does not get appended to the existing Collaboration Rules. 
	Save	Saves changes made to the Collaboration Definition (XSLT) file.
	Start Transforming	Runs the tester with the data values as shown.

If you are using the extended language options (see **Options Setup** on page 57), an *Encoding* icon (see Figure 278) appears in the Collaboration Tester. This icon allows you to specify the data encoding for the input file (applies to XML-based data only).

Figure 278 Encoding Icon

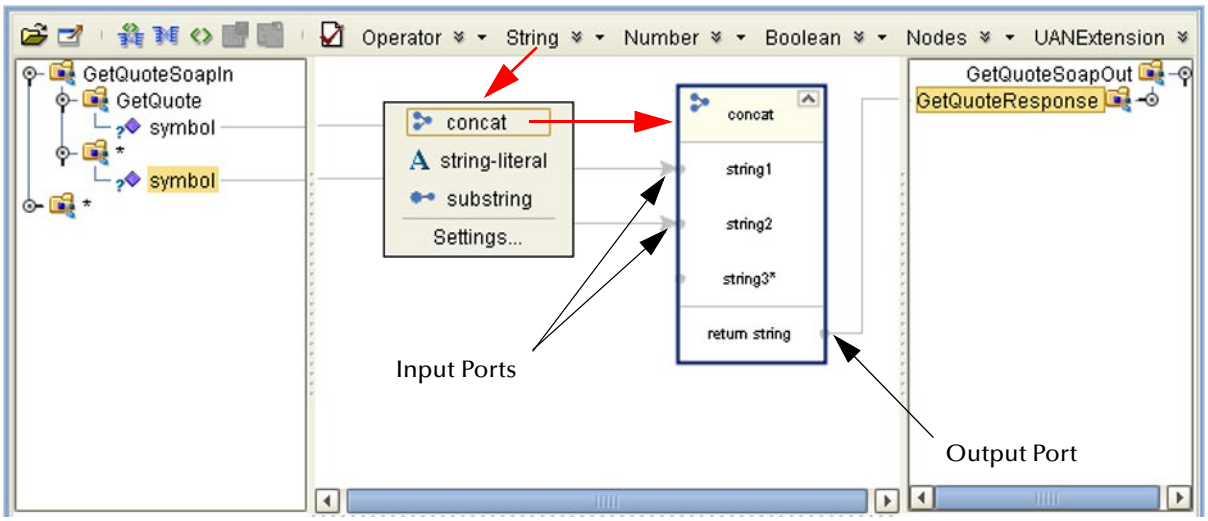
Note: *Extended language options currently apply only to Japanese and Korean localized versions of eGate Integrator.*

Collaboration Method Menus

The Collaboration Method menus list predefined methods for creating Collaboration Definitions without dealing with XSLT code (see **Predefined Collaboration Methods (XSLT)** on page 324). Clicking a category in the toolbar displays a drop-down list of methods for that category. Clicking an individual method in the list places the corresponding Method Box onto the Business Rules Designer mapping canvas.

As an example, clicking **String** in the toolbar displays a drop-down menu listing string methods. Clicking **concat** in the menu places an **concat** method box on the mapping canvas. The method box contains input and output ports which you connect to the appropriate nodes in the OTD tree structures by dragging the cursor.

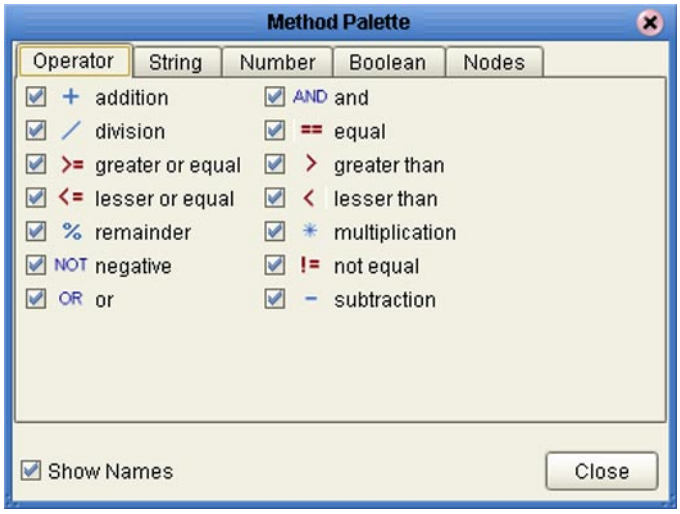
Figure 279 Collaboration Definition Editor (XSLT): Method Menus and Boxes



Clicking the **Settings** option in any menu displays the Method Palette dialog box shown in Figure 280, which allows you to select the methods that appear in the drop-down menus. Select a check box to add the method to the menu; clear a check box to remove the method from the menu.

Collaboration Method Palette (XSLT)

Figure 280 Collaboration Method Palette (XSLT)



Collaboration Method Boxes (XSLT)

The method boxes are placed on the mapping canvas of the Business Rules Designer by clicking the method in the drop-down method menu. As shown in **Figure 279 on page 322**, the method boxes typically have input and output ports that you link to fields in the left and right panels, respectively. The method boxes are expanded by default (see Figure 281); you can collapse them (see Figure 282) by clicking the caret (^) in the upper right corner of the box. Clicking the now-inverted caret expands the box. Some boxes expand further as needed to provide additional argument nodes.

Figure 281 Expanded Method Box

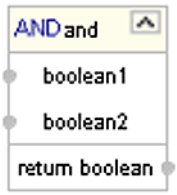


Figure 282 Collapsed Method Box

8.4 Predefined Collaboration Methods (XSLT)

8.4.1 Operator Methods

Figure 283 Operator Methods Menu

Table 83 Operator Collaboration Methods (XSLT)

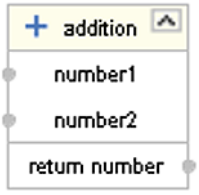
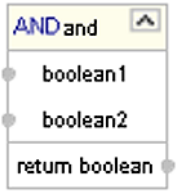
Method Box	Description/Usage
	The **addition** method adds the value of *number1* to the value of *number2*, returns the sum.
	The **and** method returns Boolean true if both *boolean1* and *boolean2* are true; otherwise, returns Boolean false.

Table 83 Operator Collaboration Methods (XSLT)

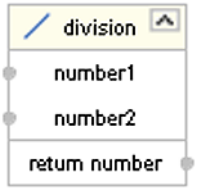
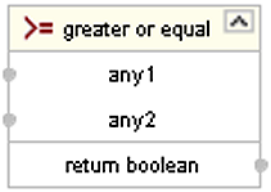
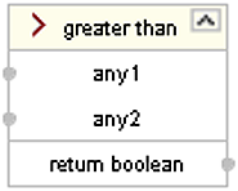
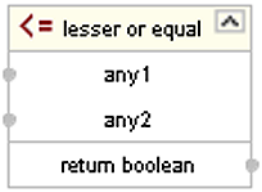
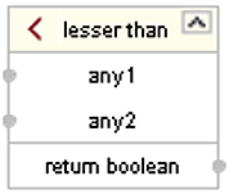
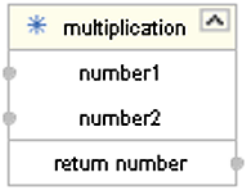
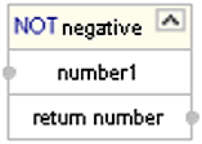
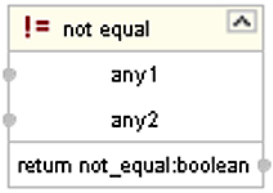
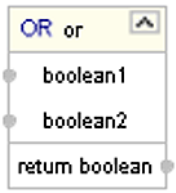
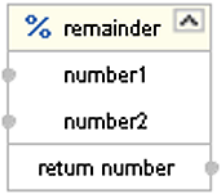
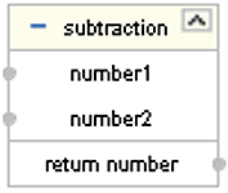
Method Box	Description/Usage
	The **division** method divides the value of *number1* by the value of *number2*, returns the quotient.
	The **equal** method returns Boolean true if *any1* is equal to *any2*; otherwise, returns Boolean false.
	The **greater_or_equal** method returns Boolean true if *any1* is greater than or equal to *any2*; otherwise, returns Boolean false.
	The **greater_than** method returns Boolean true if *any1* is greater than *any2*; otherwise, returns Boolean false.
	The **lesser_or_equal** method returns Boolean true if *any1* is less than or equal to *any2*; otherwise, returns Boolean false.
	The **lesser_than** method returns Boolean true if *any1* is less than *any2*; otherwise, returns Boolean false.

Table 83 Operator Collaboration Methods (XSLT)

Method Box	Description/Usage
	The **multiplication** method multiplies the value of *number1* by the value of *number2*, returns the product.
	The **negative** method returns the arithmetic negation of *number1*.
	The **not_equal** method returns Boolean true if *any1* is not equal to *any2*; otherwise, returns Boolean false.
	The **OR** method returns Boolean false if both *boolean1* and *boolean2* are false; otherwise, returns Boolean true.
	The **remainder** method divides the numerical value of *number1* by the numerical value of *number2*, and returns the remainder.
	The **subtraction** method subtracts the numerical value of *number2* from the numerical value of *number1*, returns the difference.

8.4.2 String Methods

Figure 284 String Methods Menu

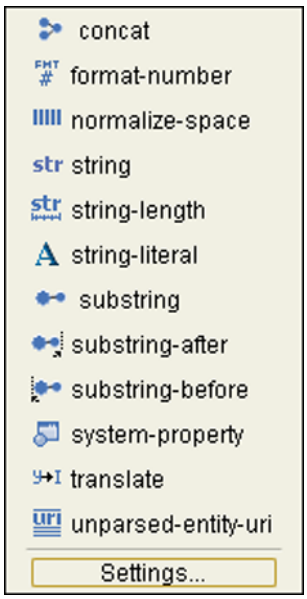


Table 84 String Collaboration Methods (XSLT)

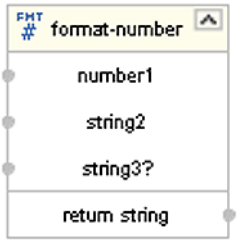
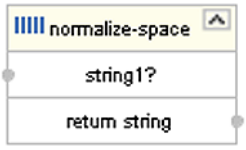
Method Box	Description/Usage
	The **concat** method returns the string created by concatenating *string2* to the end of *string1*, and *string3** (if present) to the end of *string2*.
	The **format-number** method converts its first argument (*number1*) to a string using the format pattern string specified by the second argument (*string2*) and the decimal-format named by the third argument (*string3*) or the default decimal-format, if there is no third argument.
	The **normalize-space** method returns the argument *string1?* with whitespace normalized by stripping leading and trailing whitespace and replacing sequences of whitespace characters by a single space. If the argument is omitted, it defaults to the string-value of the context node.

Table 84 String Collaboration Methods (XSLT)

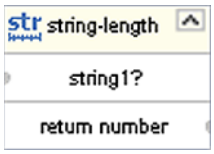
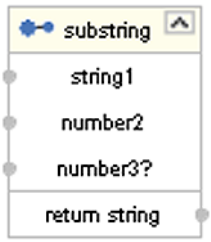
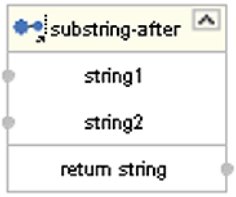
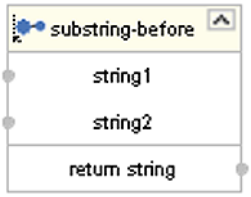
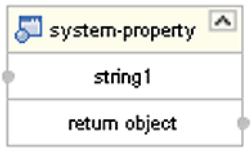
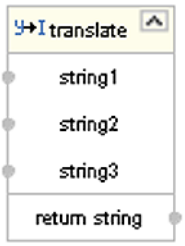
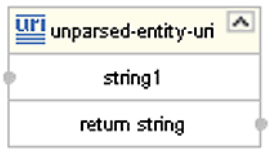
Method Box	Description/Usage
	The **string** method returns a string representation of the input object.
	The **string-length** method returns the number of characters in the string.
	Dragging the icon first displays the *String Literal* dialog box, where you enter the literal value; for example, "Data received":  The **string-literal** method then returns a string having the specified value.
	The **substring** method returns the substring of the first argument (*string1*) starting at the position specified in the second argument (*number2*) with length specified in the third argument (*number3?*). If the third argument is not specified, it returns the substring starting at the position specified in the second argument and continuing to the end of the string.
	The **substring-after** method returns the substring of the first argument (*string1*) that follows the first occurrence of the second argument (*string2*) in the first argument string. Returns an empty string if the first argument string does not contain the second argument string.
	The **substring-before** method returns the substring of the first argument (*string1*) that precedes the first occurrence of the second argument (*string2*) in the first argument string. Returns an empty string if the first argument string does not contain the second argument string.

Table 84 String Collaboration Methods (XSLT)

Method Box	Description/Usage
	The **system-property** method returns an object representing the value of the system property identified by *string1*. If there is no such system property, the empty string should be returned.
	The **translate** method returns the first argument (*string1*) with occurrences of characters in the second argument (*string2*) replaced by the character at the corresponding position in the third argument (*string3*). If a character occurs more than once in the second argument (*string2*) , then the first occurrence determines the replacement character. If the third argument (*string3*) is longer than the second argument (*string2*) , then excess characters are ignored. Refer to the W3C *XML Path Language* documentation for additional conditions.
	The **unparsed-entity-uri** method returns the URI of the unparsed entity with the specified name (*string1*) in the same document as the context node. It returns an empty string if there is no such entity.

8.4.3 Number Methods

Figure 285 Number Methods Menu

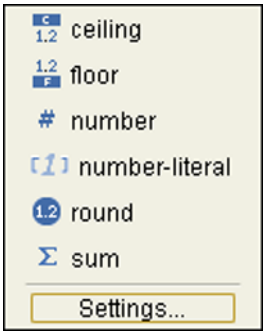


Table 85 Number Collaboration Methods (XSLT)

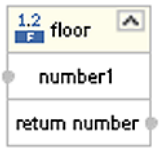
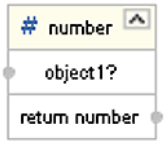
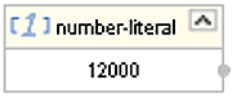
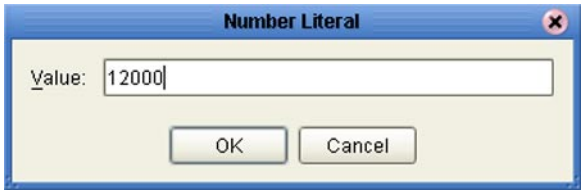
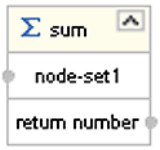
Method Box	Description/Usage
	The **ceiling** method returns the smallest (closest to negative infinity) number that is not less than the argument and that is an integer.
	The **floor** method returns the largest (closest to positive infinity) number that is not greater than the argument and that is an integer.
	The **number** method converts its argument (*object1*) to a number. An object of a type other than the four basic types is converted to a number in a way that is dependent on that type.
	Dragging the icon first displays the *Number Literal* dialog box, where you enter the literal value, such as "12000":  The **number-literal** method then returns a number having the specified value.

Table 85 Number Collaboration Methods (XSLT)

Method Box	Description/Usage
	The **round** method returns the number that is closest to the argument *number1* and that is an integer. ▪ If there are two such numbers, then the one that is closest to positive infinity is returned. ▪ If the argument is not a number (NaN), then NaN is returned. ▪ If the argument is positive infinity, then positive infinity is returned. ▪ If the argument is negative infinity, then negative infinity is returned. ▪ If the argument is positive zero, then positive zero is returned. ▪ If the argument is negative zero, then negative zero is returned. ▪ If the argument is less than zero, but greater than or equal to -0.5, then negative zero is returned.
	The **sum** method returns the sum, for each node in the argument *node-set1*, of the result of converting the string-values of the node to a number.

8.4.4 Boolean Methods

Figure 286 Boolean Methods Menu

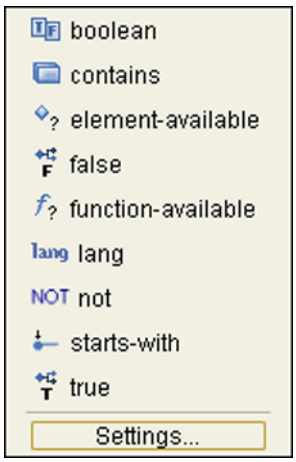


Table 86 Boolean Collaboration Methods (XSLT)

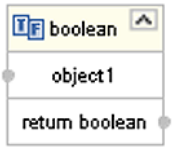
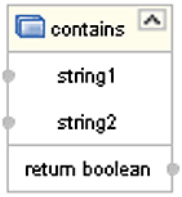
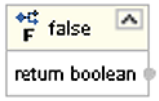
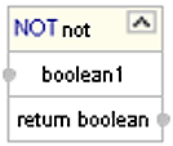
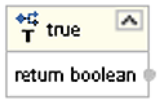
Method Box	Description/Usage
	The **boolean** method converts the argument *object1* to a Boolean true or false as follows: ▪ A number is true if and only if it is neither ±zero nor NaN (not a number). ▪ A node-set is true if and only if it is non-empty. ▪ A string is true if and only if it is non-zero. ▪ An object of a type other than the four basic types is converted to a Boolean in a way that is dependent on that type.
	The **contains** method returns Boolean **true** if the first argument (*string1*) contains the second argument (*string2*); if not, returns Boolean **false**.
	The argument *string1* must be a QName, which is expanded into an expanded-name using the namespace declarations in scope for the expression. The **element-available** method returns true if and only if the expanded-name is the name of an instruction. If the expanded-name has a namespace URI equal to the XSLT namespace URI, then it refers to an element defined by XSLT. Otherwise, it refers to an extension element. If the expanded-name has a null namespace URI, the element-available function will return false.
	The **false** method returns Boolean **false**.

Table 86 Boolean Collaboration Methods (XSLT)

Method Box	Description/Usage
	The argument *string1* must be a QName, which is expanded into an expanded-name using the namespace declarations in scope for the expression. The **function-available** method returns true if and only if the expanded-name is the name of a function in the function library. If the expanded-name has a non-null namespace URI, then it refers to an extension function; otherwise, it refers to a function defined by XPath or XSLT.
	The **lang** (**language**) method returns Boolean **true** or **false** depending upon whether the language of the context node as specified by *xml:lang* attributes is the same as, or is a sub-language of, the language specified by the argument string (*string1*). Returns Boolean **false** if the attribute *xml:lang* does not exist.
	The **not** method returns the inverse of *boolean1*.
	The **starts-with** method returns Boolean **true** if the first argument (*string1*) starts with the second argument (*string2*); if not, returns Boolean **false**.
	The **true** method returns Boolean **true**.

8.4.5 Nodes Methods

Figure 287 Nodes Methods Menu

Table 87 Nodes Collaboration Methods (XSLT)

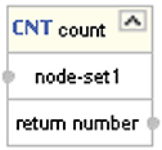
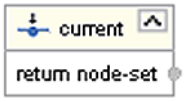
Method Box	Description/Usage
	The **count** method returns the number of nodes in the argument *node-set1*.
	The **current** method returns a node-set that has the current node as its only member.

Table 87 Nodes Collaboration Methods (XSLT)

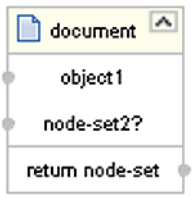
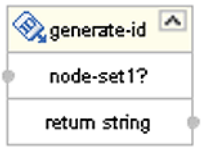
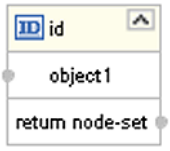
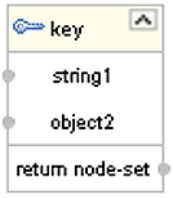
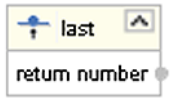
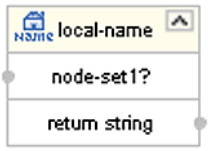
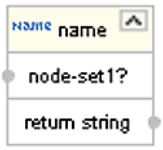
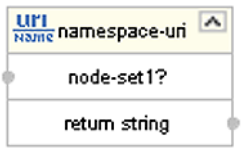
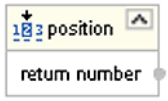
Method Box	Description/Usage
	The **document** method allows access to XML documents other than the main source document. ▪ When the method has exactly one argument and the argument is a node-set, then for each node in the argument node-set, the result is the union of the result of calling the document function with the first argument being the string-value of the node, and the second argument being a node-set with the node as its only member. ▪ When the method has two arguments and the first argument is a node-set, then for each node in the argument node-set, the result is the union of the result of calling the document function with the first argument being the string-value of the node, and with the second argument being the second argument passed to the document function. When the first argument is not a node-set, the first argument is converted to a string as if by a call to the string function. This string is treated as a URI reference; the resource identified by the URI is retrieved.
	The **generate-id** method returns a string that uniquely identifies the node in the argument *node-set1?* that is first in document order. The unique identifier must consist of ASCII alphanumeric characters and must start with an alphabetic character. Thus, the string is syntactically an XML name.
	When the argument *object1* is not of type node-set, the **id** method converts the argument to a string as if by a call to the string function; the string is split into a whitespace-separated list of tokens; the result is a node-set containing the elements in the same document as the context node that have a unique ID equal to any of the tokens in the list.
	The **key** method does for keys what the id function does for IDs. The value of the first argument (*string1*), which specifies the name of the key, must be a QName—which is expanded into an expanded-name using the namespace declarations in scope for the expression. ▪ When the second argument is of type node-set, then the result is the union of the result of applying the key function to the string value of each of the nodes in the argument node-set. ▪ When the second argument is of any other type, the argument is converted to a string as if by a call to the string function; it returns a node-set containing the nodes in the same document as the context node that have a value for the named key equal to this string.
	The **last** method returns a number equal to the context size from the expression evaluation context.

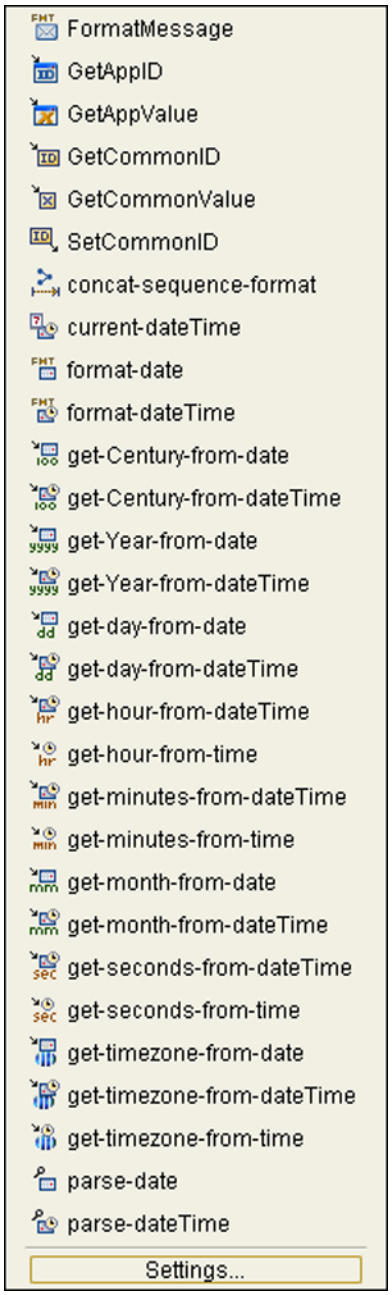
Table 87 Nodes Collaboration Methods (XSLT)

Method Box	Description/Usage
	The **local-name** method returns the local part of the expanded-name of the node in the argument *node-set1?* that is first in document order. If the argument is empty or the first node has no expanded-name, an empty string is returned. If the argument is omitted, it defaults to a node-set with the context node as its only member.
	The **name** method returns a string containing a QName representing the expanded-name of the node in the argument *node-set1?* that is first in document order. If the argument is empty or the first node has no expanded-name, an empty string is returned. If the argument it omitted, it defaults to a node-set with the context node as its only member.
	The **namespace-uri** method returns the namespace URI of the expanded-name of the node in the argument *node-set1?* that is first in document order. If the argument is empty, the first node has no expanded-name, or the namespace URI of the expanded-name is null, an empty string is returned. If the argument is omitted, it defaults to a node-set with the context node as its only member.
	The **position** method returns a number equal to the context position from the expression evaluation context.

8.4.6 UAN Extension Methods

These methods are used in conjunction with the Siebel UAN eWay for implementing UAN Extension functions during run time. Employing these methods causes the eWay to interact with the cross-reference (XRef) database to perform the specified request. Since these are run-time functions, their operation cannot be tested in the XSLT Tester. Definitions of the parameters used in these methods are listed in Table 89.

Figure 288 UAN Extension Methods Menu



Note: This menu and the associated methods appear only if you have installed the SeeBeyond UANExtension add-on.

Table 88 UAN Extension Collaboration Methods (XSLT)

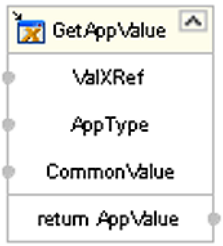
Method Box	Description/Usage
	The **formatMessage** method function serves as a message packaging service in UAN. It is able to return a formatted message strings based on specifications given by the caller of the function. This function is able to retrieve pre-defined message text in repository, perform argument substitution and argument cross-referencing, and return the packaged message in the specified language. Only the **Code** and **Lang** parameters are required; all others are optional. ▪ **Code** is the code of the desired message; it is used to locate the desired message definition in the repository ▪ **Lang** is the desired language text of the message. ▪ **AppInst** is the name of the application instance that is to receive the message. Passing a null value indicates id cross-referencing will not be performed. ▪ **AppType** is the type of the application that is to receive this message. Passing a null value indicates value cross-referencing will not be performed. ▪ **ArgN** is the value of the Nth argument.
	The **getAppID** method is used to obtain a specific application ID by giving the ID cross reference name (**IDXRef**), application instance name (**AppInst**) and common ID (**commonID**). The return value of this method is the ID for the target application instance. If the application ID has not been registered, "" is returned.
	The **getAppValue** method returns the application value (**AppValue**) corresponding to the specified common value (**CommonValue**), given the application type (**AppType**) and type of object being cross-referenced (**ValXRef**).
	The **getCommonID** method is used to obtain the common ID corresponding to a specific application ID by giving the ID cross reference name (**IDXRef**), application instance name (**AppInst**) and application ID (**AppID**). The return value of this method is the common ID for the target application instance. If the application ID has not been registered, "" is returned.

Table 88 UAN Extension Collaboration Methods (XSLT)

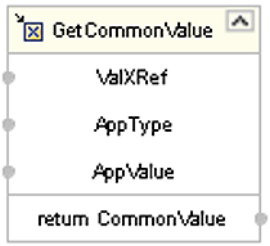
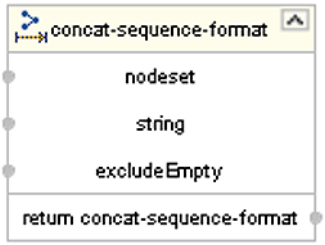
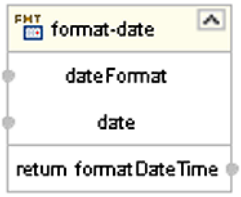
Method Box	Description/Usage
	The **getCommonValue** method returns the common value (**CommonValue**) corresponding to the specified application value (**AppValue**), given the application type (**AppType**) and type of object being cross-referenced (**ValXRef**).
	The **setCommonID** method is used to create, query, and establish the relationship between application ID (**AppID**) and common ID (**CommonID**). ▪ **To Create:** If an application ID is being registered for the first time, the system will generate a new common ID to store the information. ▪ **To Query:** If an application ID is already registered, the common ID will be returned. ▪ **To Establish:** If the **CommonID** parameter is specified, the relationship between this common ID and the application ID will be established, linking the current application with an application ID. ▪ If the **CommonID** parameter is not specified, this method should generate an unique ID or return the existing common ID.
	The **concat-sequence-format** method concatenates the string value of each node in order, using a string separator between entries. If the Boolean parameter **excludeEmpty** is *true*, empty strings are not included. If **excludeEmpty** is omitted, *false* is assumed.
	The **currentDateTime** method returns the current time as a **dateTime** string.
	The **formatDate** method takes the given **date** string and converts it to an ISO 8601 **dateTime** string as specified by the **dateFormat** string.

Table 88 UAN Extension Collaboration Methods (XSLT)

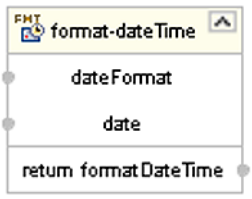
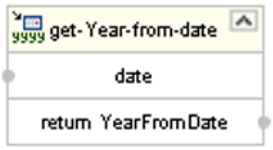
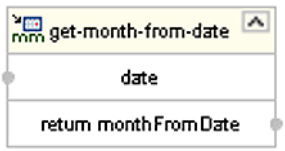
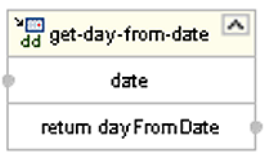
Method Box	Description/Usage
	The **formatDateTime** method takes an ISO 8601 **dateTime** string and converts it to a **dateTime** string as specified by the **dateFormat** string.
	The **getCenturyFromDate** method extracts the century as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.
	The **getCenturyFromDateTime** method extracts the century as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getYearFromDate** method extracts the year as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.
	The **getYearFromDateTime** method extracts the year as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getMonthFromDate** method extracts the month as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.
	The **getMonthFromDateTime** method extracts the month as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getDayFromDate** method extracts the day as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.

Table 88 UAN Extension Collaboration Methods (XSLT)

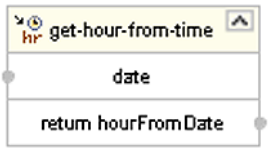
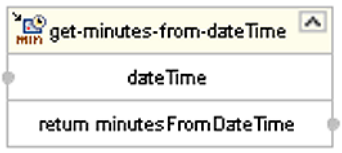
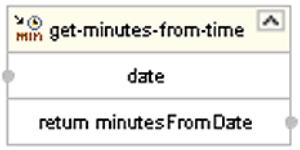
Method Box	Description/Usage
	The **getDayFromDateTime** method extracts the day as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getHourFromDateTime** method extracts the hour as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone. The hour value ranges from 0 to 23, inclusive.
	The **getHourFromTime** method extracts the hour as a number from a string with XML Schema **time** datatype, which is the left-truncated lexical representation for **dateTime** (hh:mm:ss.ssss) with optional timezone. The hour value ranges from 0 to 23, inclusive.
	The **getMinutesFromDateTime** method extracts the minutes as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone. The minutes value ranges from 0 to 59, inclusive.
	The **getMinutesFromTime** method extracts the minutes as a number from a string with XML Schema **time** datatype, which is the left-truncated lexical representation for **dateTime** (hh:mm:ss.ssss) with optional timezone. The minutes value ranges from 0 to 59, inclusive.
	The **getSecondsFromDateTime** method extracts the seconds as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone. The seconds value ranges from 0 to 59, inclusive.
	The **getSecondsFromTime** method extracts the seconds as a number from a string with XML Schema **time** datatype, which is the left-truncated lexical representation for **dateTime** (hh:mm:ss.ssss) with optional timezone. The seconds value ranges from 0 to 59, inclusive.
	The **getTimezoneFromDate** method extracts the timezone as a number from a string having an XML Schema **date** datatype, the encoding of which is based on the ISO 8601 reduced format (CCYY-MM-DD) with optional timezone.

Table 88 UAN Extension Collaboration Methods (XSLT)

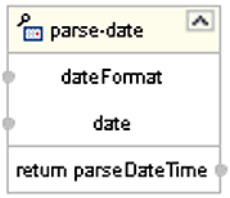
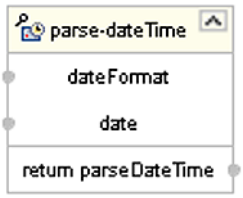
Method Box	Description/Usage
	The **getTimezoneFromDateTime** method extracts the timezone as a number from a string having an XML Schema **dateTime** datatype, the encoding of which is based on the ISO 8601 extended format (CCYY-MM-DDThh:mm:ss) with optional timezone.
	The **getTimezoneFromTime** method extracts the timezone as a number from a string with XML Schema **time** datatype, which is the left-truncated lexical representation for **dateTime** (hh:mm:ss.ssss) with optional timezone.
	The **parseDate** method parses the given **dateValue** string into an ISO 8601 **date** string as specified by the **dateFormat** string.
	The **parseDateTime** method parses the given **date** string into an ISO 8601 **dateTime** string as specified by the **dateFormat** string.

Table 89 UAN Extension Collaboration Method Parameter Definitions

Parameter	Description/Usage
AppID	The key, having more than 50 characters, that is used to identify the object in the particular application.
AppInst	A string of no more than 50 characters that uniquely identifies the application instance.
AppType	A string of no more than 50 characters that uniquely identifies the application type.
AppValue	The value, having no more than 50 characters, that is used by the application type.
CommonID	The key, having no more than 50 characters, that is to be used in the neutral, common format.
CommonValue	The value, having no more than 50 characters, that is to be used in the neutral, common format.
IDXRef	A string of no more than 50 characters that identifies the type of object within the system that is being cross-referenced.
ValueXRef	A string of no more than 50 characters that identifies the type of object within the application that is being cross-referenced.

Chapter 9

Environments

This chapter describes the process of defining eGate Integrator run-time Environments, and the various components of an Environment.

What's in This Chapter

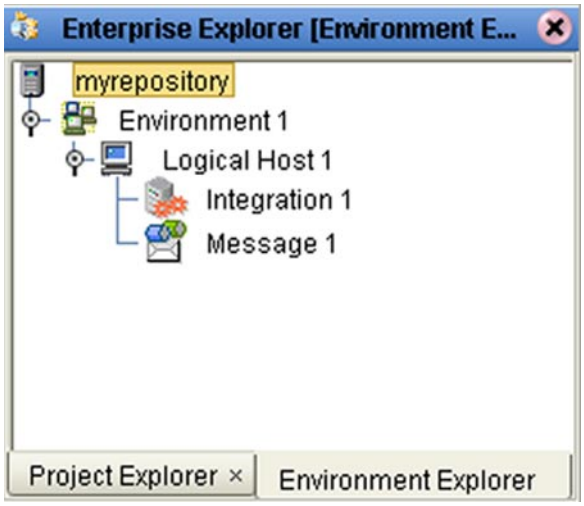
9.1 Overview

Projects are run within *Logical Hosts*, which contain the logical resources required by the Project at run time, including integration servers and message servers. The Logical Hosts, in turn, are defined within *Environments*, which represent the physical resources required to implement the Project. The Environment also contains information about external systems with which the ICAN Project interacts.

9.2 Environment Explorer

The **Environment Explorer** displays the contents of the Repository that belong to the selected Environment (see Figure 289).

Figure 289 Enterprise Explorer: Environment Explorer View



The Environment Explorer is used in conjunction with the Environment Editor to create and configure the components of a run-time Environment. Each component in the Environment Explorer has an icon to identify the component type (see **Environment Explorer Icons** on page 346). Right-clicking on a component displays a context menu for that component (see **Context Menus** on page 347), from which you can select appropriate actions.

9.2.1 Environment Explorer Icons

The icons described in Table 90 appear in the Environment Explorer.

Table 90 Environment Icons

Icon	Function
	Represents the **Repository**, which is the central ICAN database where all Project information is saved. Binary files required at run time are also stored here.
	Represents a run-time **Environment**, which contains Logical Hosts and information about external systems.
	Represents a **Logical Host**, which contains the various logical components and files that are required at run time.
	Represents an **Environmental Constant**, which you can use to automate eWay and message destination configuration changes.
	Represents a **Scheduler** component of an Environment, which you can use to set data transfer to occur at set intervals.
	Represents a **SeeBeyond Integration Server** or third-party **application server**, which manages the Collaborations and other process interactions of an ICAN Project. The integration server is deployed to a Logical Host.
	Represents a **SeeBeyond JMS IQ Manager** or third-party **message server**, which is used to store and forward eGate Integrator system messages. The message server is deployed to a Logical Host.

9.2.2 Context Menus

Right-clicking on a component in the Environment Explorer displays a context menu for that component. Included here are descriptions of options for the following component context menus:

- **Repository Menu** on page 347
- **Environment Menu** on page 348
- **Logical Host Menu** on page 349

Repository Menu

Figure 290 Repository Menu

Table 91 Repository Menu Options

Option	Function
New Environment	Displays a dialog box with which you can create a new Environment.
Configure SNMP Agent	Displays a dialog box with which you can modify the SNMP agent properties.
Save Changes to Repository	Saves any changes you have made in the Environment Editor to the Repository.
Refresh All from Repository	Refreshes the Environment Explorer to display the current contents of the Repository. (Open editors are not refreshed.)

Environment Menu

Figure 291 Environment Menu

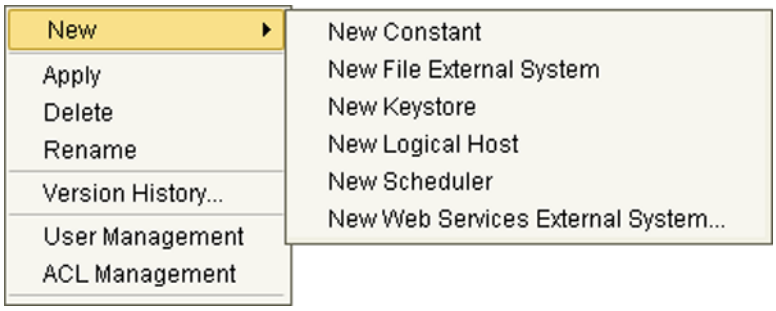


Table 92 Environment Menu Options

Option		Function
New	Constant	Displays a dialog box with which you can add a constant to the selected Environment. See **Defining Environmental Constants** on page 353.
	File External System	This option is present only when the File eWay is installed. Other installed eWays also appear as similar options.
	Keystore	Adds a new keystore to the selected Environment.
	Logical Host	Adds a new Logical Host to the selected Environment.
	Scheduler	Displays a dialog box with which you can add a new scheduling component to the selected Environment.
	Web Services External System	Adds a Web Services External System to the selected Environment. See **Web Services External System** on page 418.
Apply		Applies the latest configuration in the Repository to the selected Environment.
Delete		If you have *delete* privileges for the Environment (see ACL Management, below), a dialog box is displayed in which you confirm that you want to delete the selected Environment. Clicking **Yes** then deletes the Environment.
Rename		Activates the field, allowing you to rename the selected Environment.
Version History		Displays a dialog box with which you can track the version history for the selected Environment. See **Viewing a Component's Version History** on page 80 for more information.
User Management		Displays a dialog box with which an Administrator can manage message server access. See the *eGate Integrator System Administration Guide*.

Table 92 Environment Menu Options

Option		Function
ACL Management		Displays the **ACL Properties** dialog box, with which an Administrator can assign read/write/delete privileges to users for the selected Environment. See the *eGate Integrator System Administration Guide*.

Logical Host Menu

Figure 292 Logical Host Menu

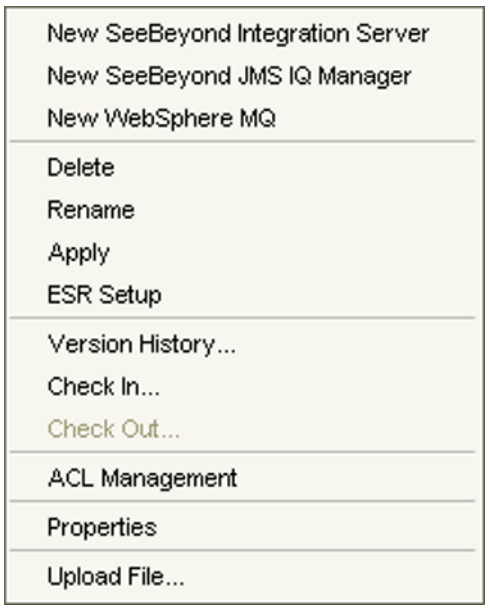


Table 93 Logical Host Menu Options

Option	Function
New SeeBeyond Integration Server	Adds a new SeeBeyond Integration Server to the selected Logical Host.
New SeeBeyond JMS IQ Manager	Adds a new SeeBeyond JMS IQ Manager to the selected Logical Host. See the *eGate Integrator JMS Reference Guide* for details.
New WebSphere MQ	Adds a new IBM WebSphere MQ message server to the selected Logical Host. See the *eGate Integrator JMS Reference Guide* for details.
Delete	Deletes the selected Logical Host, subject to the following conditions: ▪ You have *delete* privileges for the Logical Host (see ACL Management, below). ▪ The Logical Host is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected Logical Host. Clicking **Yes** then deletes the Logical Host.

Table 93 Logical Host Menu Options

Option	Function
Rename	Activates the field, allowing you to rename the selected Logical Host.
Apply	Applies the latest configuration in the Repository to the selected Logical Host.
ESR Setup	Displays a dialog box with which an Administrator can select emergency software releases (ESRs) to add to the selected Logical Host.
Version History	Displays a dialog box with which you can track the version history for the selected Logical Host. See **Viewing a Component's Version History** on page 80 for more information.
Check In	Displays a dialog box with which you can check in a new version of a Logical Host. Refer to **Checking a Component In** on page 81 for more details.
Check Out	Displays a dialog box with which you can check out the current version of a Logical Host. See **Checking a Component Out** on page 83 for more information.
ACL Management	Displays the **ACL Properties** dialog box, with which an administrator can assign read/write/delete privileges to users for the selected Logical Host. See the *eGate Integrator System Administration Guide*.
Properties	Displays a dialog box with which you can modify the configuration properties for the selected Logical Host.
Upload File	Displays a dialog box with which you can upload third-party libraries (**.jar** files) to the Logical Host.

Note: *If you are using BEA WebLogic and/or IBM WebSphere, the Application Servers and JMS Message Servers for these products will also appear in the context menu (see Figure 293).*

Figure 293 Logical Host Menu with Third-Party Servers

Integration Server

The menu shown in Figure 294 is used for both the SeeBeyond Integration Server and third-party application servers. The configuration properties, however, are different.

Figure 294 Integration Server Menu

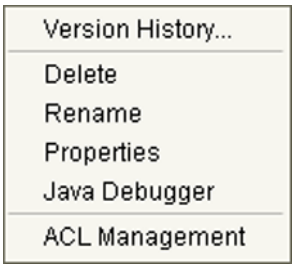


Table 94 Integration Server Menu Options

Option	Function
Version History	Displays a dialog box with which you can track the version history for the selected integration server. See **Viewing a Component's Version History** on page 80 for more information.
Delete	Deletes the selected integration server, subject to the following conditions: ▪ You have *delete* privileges for the integration server (see ACL Management, below). ▪ The integration server is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected integration server. Clicking **Yes** then deletes the integration server.
Rename	Activates the field, allowing you to rename the selected integration server.
Properties	Displays a dialog box with which you can modify the configuration properties for the selected integration server.
Java Debugger	Displays the Java Debugger (see **Invoking the Java Debugger** on page 300).
ACL Management	Displays the **ACL Properties** dialog box, with which an administrator can assign read/write/delete privileges to users for the selected integration server. See the *eGate Integrator System Administration Guide*.

SeeBeyond JMS IQ Manager

The menu shown in Figure 295 is used for both the SeeBeyond JMS IQ Manager and third-party message servers. The configuration properties, however, are different.

Figure 295 JMS IQ Manager Menu

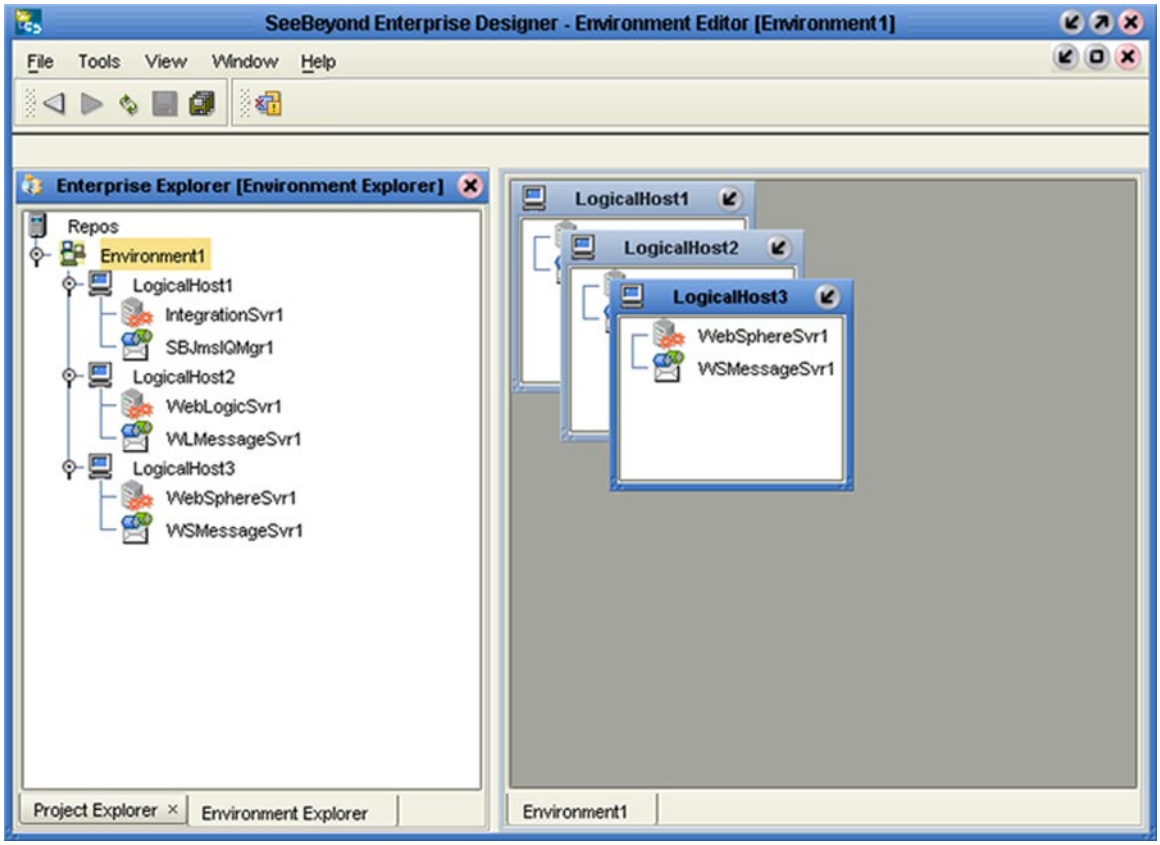
Table 95 Integration Server Menu Options

Option	Function
Delete	Deletes the selected message server, subject to the following conditions: ▪ You have *delete* privileges for the message server (see ACL Management, below). ▪ The message server is not checked out by anyone other than yourself. If these conditions are true, a dialog box is displayed in which you confirm that you want to delete the selected message server. Clicking **Yes** then deletes the message server.
Rename	Activates the field, allowing you to rename the selected message server.
Properties	Displays a dialog box with which you can modify the configuration properties for the selected message server.
Version History	Displays a dialog box with which you can track the version history for the selected message server. See **Viewing a Component's Version History** on page 80 for more information.
ACL Management	Displays the **ACL Properties** dialog box, with which an administrator can assign read/write/delete privileges to users for the selected message server. See the *eGate Integrator System Administration Guide*.

9.3 Environment Editor

Clicking an Environment icon in the Environment Explorer invokes the Environment Editor, which provides a canvas in which you can create and customize an Environment (see Figure 296).

Figure 296 Environment Editor

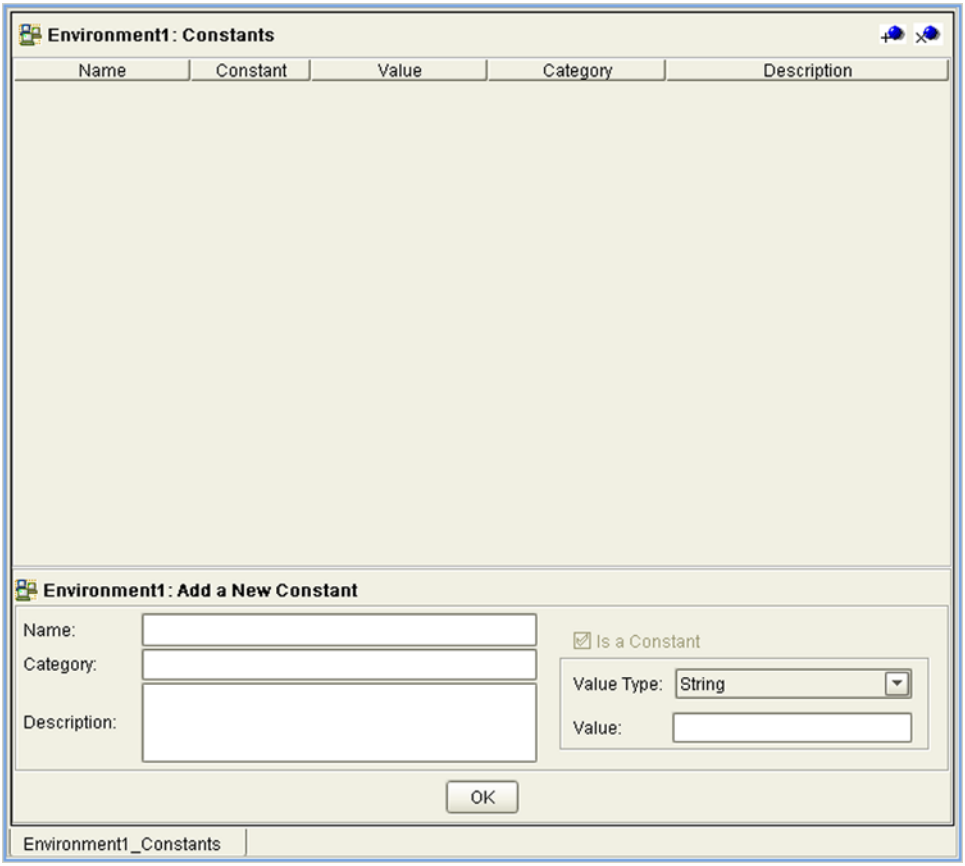


Here you can see the various components (Logical Hosts, servers, and external systems) included in the selected Environment. New Environments are added through the use of the Repository context menu (see **Repository Menu** on page 347). Components are added to the Environment by selecting options in the Environment and Logical Host context menus (see **Environment Menu** on page 348 and **Logical Host Menu** on page 349, respectively).

9.3.1 Defining Environmental Constants

Environmental constants are name/value pairs that are visible across the Environment. Selecting the **New Constant** option from the Environment context menu displays the Constants panel in the Environment Editor (see Figure 297).

Figure 297 Environmental Constants Panel



All constants defined for the specific Environment are listed in the *Constants* section of the panel, along with their various properties. New constants are added using the *Add a New Constant* section of the panel.

Note: *When you create an Environmental constant, you assign a permanent value to it—which cannot be overridden.*

Table 96 Environmental Constants Panel Icons

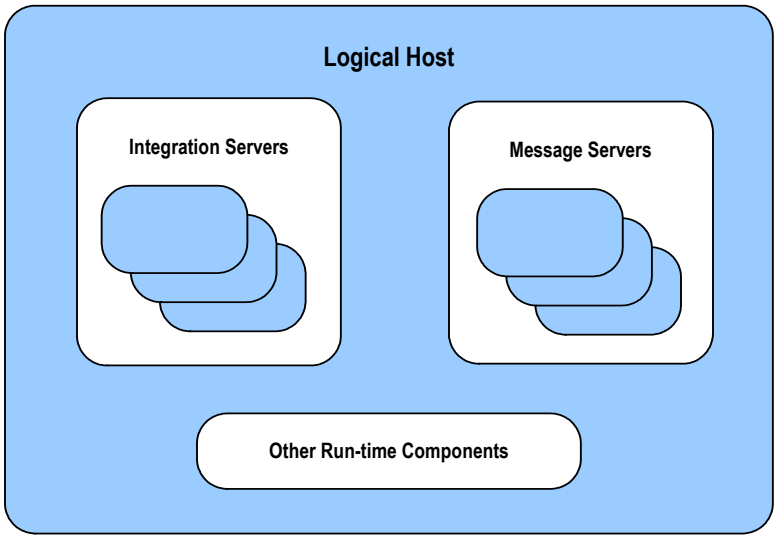
Icon	Name	Function
	Add a New Constant	Adds a new constant to the list.
	Delete a Highlighted Constant	Deletes the selected constant from the list.

9.4 Logical Hosts

9.4.1 Overview

A Logical Host contains the eGate Integrator run-time components that are installed on a host hardware platform. A Logical Host can be a member of only one Environment, but each Environment can contain multiple Logical Hosts. Both integration servers and message servers are deployed to the Logical Host, as illustrated in Figure 298.

Figure 298 Logical Hosts



The master service of the Logical host is the Management Agent. This service starts the other services on the Logical Host as part of the bootstrap process. The Management Agent also communicates with the Enterprise Manager via JMX (Java Management Extensions) to report the status of the message servers and integration servers. You can view the status of these components by means of the Enterprise Monitor.

The Enterprise Monitor also allows you to control the components within the Logical Host. Stopping the Logical Host through the Enterprise Monitor stops all components except the Management Agent, which remains running so that the components can be restarted from the Enterprise Monitor. This behavior is necessary to accommodate situations where the Logical Host and Repository are located on separate computers. See the *eGate Integrator System Administration Guide* for additional information.

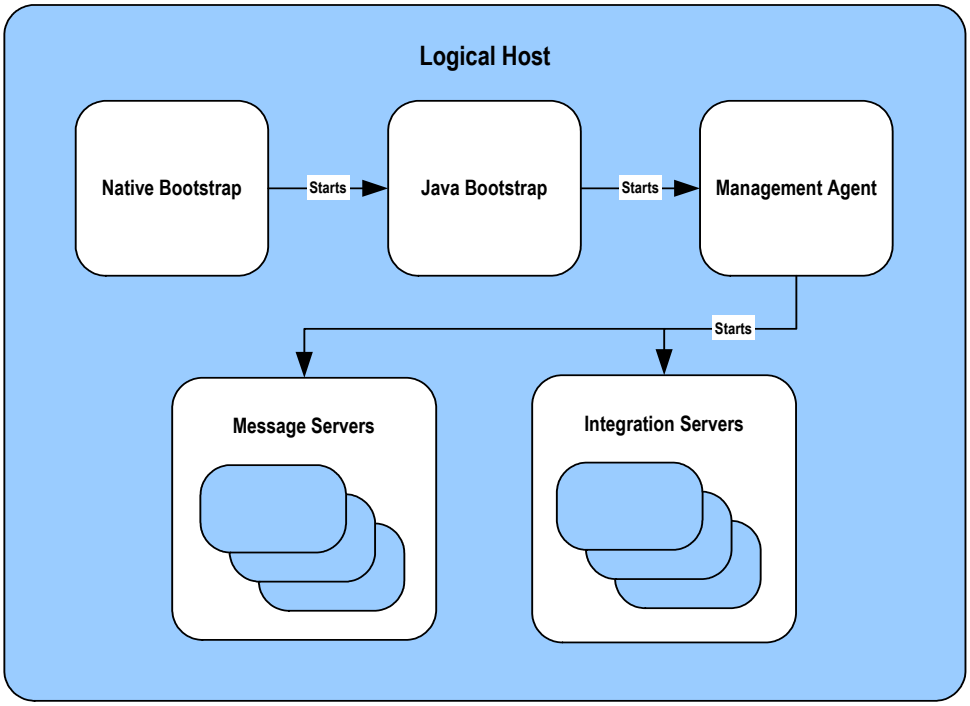
9.4.2 Bootstrapping

At run time, a platform-specific bootstrap script starts a Java bootstrap program. The first time the bootstrap runs, it downloads the Management Agent, message server, and integration server from the Repository. The Management Agent is then started, which in turn starts the message server(s) and integration server(s). Figure 299 illustrates this sequence.

Subsequent bootstraps simply start the existing Management Agent without downloading component configurations from the Repository, unless you force a download by:

- Selecting **Apply** from the Logical Host context menu (see **Logical Host Menu** on page 349).
- Using the command-line option (**-f**). See the *eGate Integrator System Administration Guide* for information.

Figure 299 Startup Sequence



Each Logical Host has a separate bootstrap process. The process is started from a script (***ICAN-root***\logicalhost\bootstrap\bin\bootstrap.bat) or (***ICAN-root***/logicalhost/bootstrap/bin/bootstrap.sh). The Repository location is furnished either by command-line parameters or the configuration properties file (***ICAN-root***\logicalhost\bootstrap\config\logical-host.properties). See the *eGate Integrator System Administration Guide* for additional information.

9.4.3 Configuring a Logical Host

Note: *Unlike changes to Project-related configuration properties, changes to Environment-related properties do not require redeployment, simply application.*

To access the configuration properties for a Logical Host

1. Right-click a Logical Host in the Environment Explorer tree to display the context menu for that Logical Host instance.
2. Select **Properties** from the context menu to display the **Properties** dialog box.
3. Select the **Logical Host Configuration** node in the properties tree to display the Logical Host Configuration Section, which contains the top-level configuration properties for the Logical Host (see Figure 300).

Figure 300 Logical Host Configuration Properties

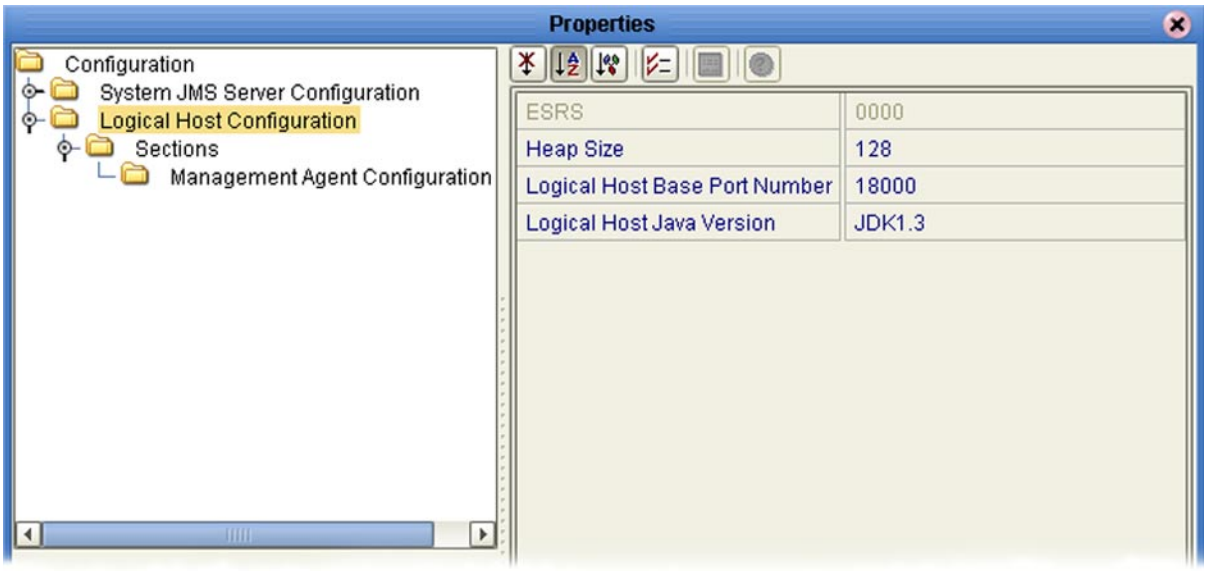


Table 97 Logical Host Configuration Properties List

Property	Description
ESRS	Shows a list of all installed Emergency Software Releases (ESRs).
Heap Size	Specifies the Heap size in Megabytes; the minimum size is **128** Mb, which is the default value. Note that this property is only for the bootstrap and management processes, and does not affect the integration server or any runtime components that are processing data.
Logical Host Base Port Number	Specifies the base port number for the Logical Host. The default value is **18000**. When multiple Logical Hosts reside on a single hardware platform, you must configure the base port numbers (see the following section).

Table 97 Logical Host Configuration Properties List

Property	Description
Logical Host Java Version	Specifies the Java version being used to the eWay RAR file generation program, so that any generated file will be properly compatible. The default value is **JDK1.3**.

Configuring the Base Port Number

If multiple Logical Hosts concurrently run on the same computer, you must ensure that each Logical Host has a different base port number to avoid conflicts. This base port number is propagated throughout the Logical Host, so that the various components are automatically given successive port numbers following the number assigned to the Logical Host itself.

The number of port numbers used in a Logical Host varies according to the specific implementation, so when assigning new base port numbers you need to skip successive numbers by an adequate amount. The default base port number is 18000, so base port numbers of 19000, 20000, and so on are recommended.

If you need to assign a specific port number to a particular Logical Host component, the automatic numbering process will skip the component port number you have assigned manually (*be sure this port number is not used elsewhere*).

Note: *While Windows accepts port numbers below 12000, UNIX does not.*

Figure 301 Management Agent Configuration Properties

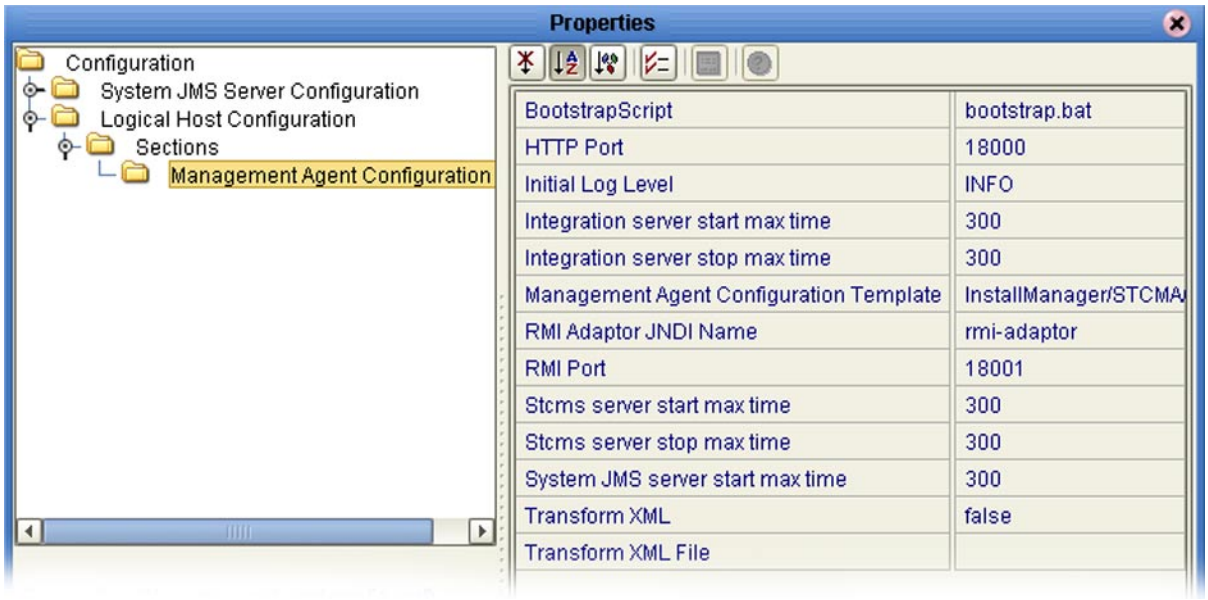


Table 98 Management Agent Configuration Properties List

Property	Description
BootstrapScript	Name of the bootstrap script; the default is **bootstrap.bat**.
HTTP Port	The HTTP port; the default value is **18000**.
Initial Log Level	The initial log level (OFF, ALL, DEBUG, INFO, WARN, ERROR, FATAL); the default value is **INFO**. See the *eGate Integrator System Administration Guide* for log level details.
Integration server start max time	The maximum time period (in seconds) during which the Management Agent will attempt to start an Integration Server. The default value is **300** seconds.
Integration server stop max time	The maximum time period (in seconds) during which the Management Agent will attempt to stop an Integration Server. The default value is **300** seconds.
Management Agent Configuration Template	The name and path of the configuration template; the default value is **InstallManager/STCMA/common/config/templates/ManagementAgent-config.vm**.
RMI Adaptor JNDI Name	The Java Naming and Directory Interface (JNDI) name of the Remote Method Invocation (RMI) adapter; the default value is **rmi-adaptor**.
RMI Port	The Remote Method Invocation (RMI) port. The displayed value depends upon the number of the Logical Host base port; the default value is **18001**.
Stcms server start max time	The maximum amount of time (in seconds) that the Management Agent will attempt to start a SeeBeyond JMS IQ Manager. The default value is **300** seconds.
Stcms server stop max time	The maximum amount of time (in seconds) that the Management Agent will attempt to stop a SeeBeyond JMS IQ Manager. The default value is **300** seconds.
System JMS server start max time	The maximum amount of time (in seconds) that the Management Agent will attempt to start the message server used internally by the ICAN Suite. The default value is **300** seconds.
Transform XML	The default value is **false**, and should not be changed.
Transform XML File	This property is not used.

See the *eGate Integrator System Administration Guide* for additional information.

9.5 Integration Servers

The Logical Host contains one or more instances of a J2EE-compatible integration server. This server is the engine that runs Collaborations for processing business logic, and eWays for communicating with external applications. The integration server provides services for security, transactions, business rules execution, and connectivity management. eGate Integrator contains the SeeBeyond Integration Server, and also supports the use of third-party application servers such as BEA WebLogic and IBM WebSphere (see **Deploying Projects to Third-Party Servers** on page 406).

9.5.1 Configuring an Integration Server

Note: *Unlike changes to Project-related configuration properties, changes to Environment-related properties do not require redeployment, simply application.*

To access the configuration properties for an integration server

1. Right-click an integration server in the Environment Explorer tree to display the context menu for that instance.
2. Select **Properties** from the context menu to display the **Properties** dialog box.
3. Select the **IS Configuration** node in the properties tree to display the top-level IS configuration properties (see Figure 302).

Figure 302 Top-level IS Configuration Properties

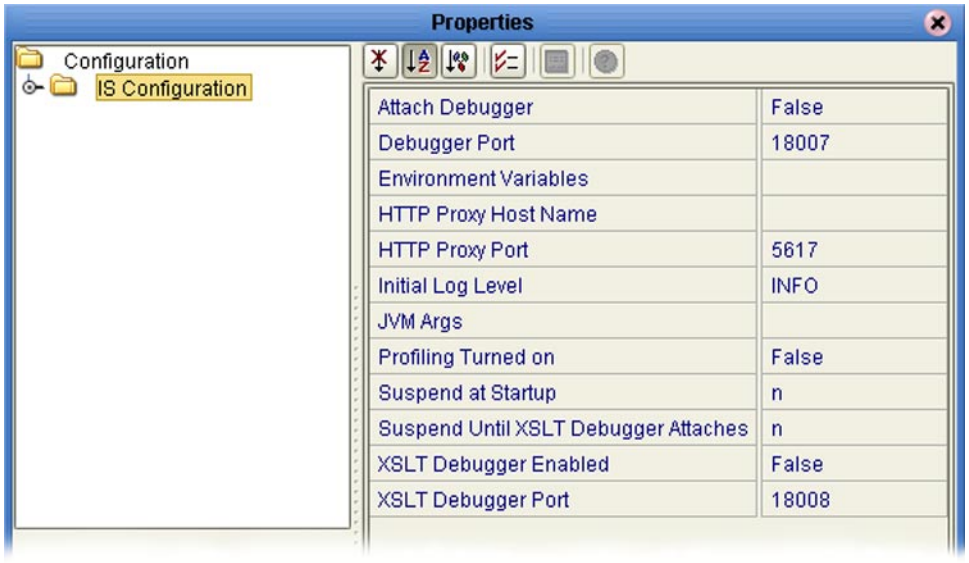


Table 99 Top-level IS Configuration Properties List

Property	Description
Attach Debugger	Enables/disables debugging for the IS. The default is **False** (disabled).
Debugger Port	This property is used only when the Debugger is enabled. The port numbers are dynamically assigned; the number shown is typical.
Environment Variables	Specifies user-defined Environment Variables. Entries here override the system settings—so if *any* variables are specified, then *all* required variables must be specified. Each element has the format **name=value**. There is no default.
HTTP Proxy Host Name	Name of the HTTP proxy host, if any. There is no default. Note: Do not use underscore character in the host name.
HTTP Proxy Port	Port for the HTTP proxy host, if any. The default is **5617**.
Initial Log Level	The initial log level (OFF, ALL, DEBUG, INFO, WARN, ERROR, FATAL); the default value is **INFO**. See the *eGate Integrator System Administration Guide* for log level details.
JVM Args	Java Virtual Machine (JVM) arguments. Each element in the collection should specify one, and only one, argument.
Profiling Turned On	Enables/disables performance monitoring for the IS. The default is **False** (disabled). To enable, change to **True** (as shown) and configure the properties described in **Performance Monitoring (Profiling) Configuration** on page 365.
Suspend at Startup	Allows the VM to begin executing before the debugger application attaches. The default is **n** (do not suspend).
Suspend Until XSLT Debugger Attaches	Allows the VM to begin executing before the XSLT debugger application attaches. The default is **n** (do not suspend).
XSLT Debugger Enabled	Enables/disables XSLT debugging for the IS. The default is **False** (disabled).
XSLT Debugger Port	This property is used only when the XSLT Debugger is enabled. The port numbers are dynamically assigned; the number shown is typical.

The IS Configuration node contains several sections, each containing detailed configuration properties for a particular IS component (including the integration server itself). You can also access these properties by selecting **Properties** from the context menus for the appropriate nodes.

Web Container Configuration

Properties included here are used for setting up Web Services.

Figure 303 Web Container Configuration Properties

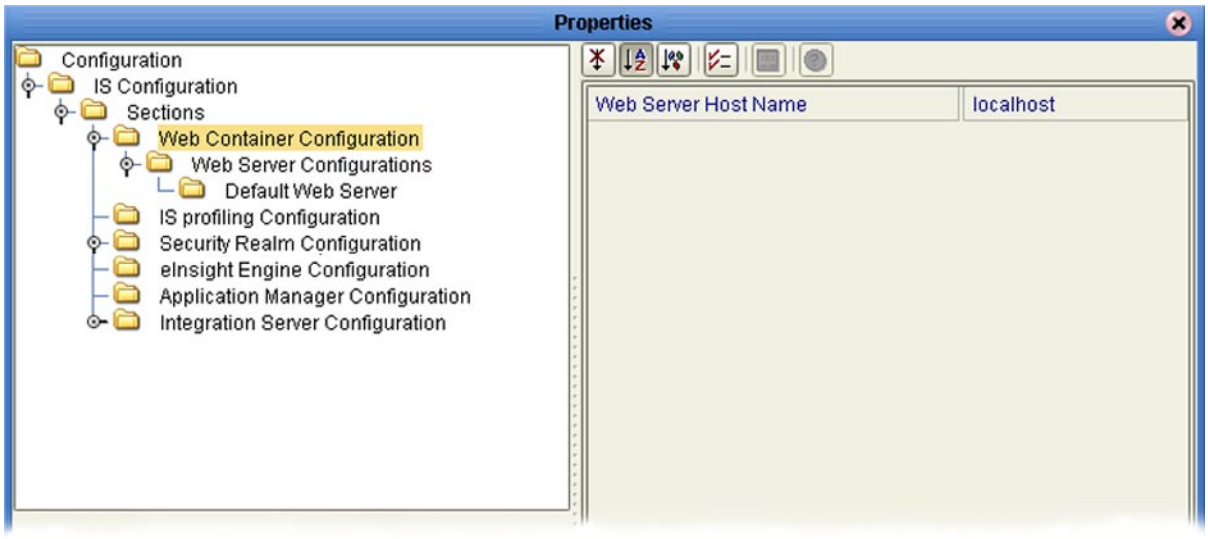


Table 100 Web Container Configuration Properties List

Property	Description
Web Server Host Name	Specifies the host name; the default is **localhost**. Note: Do not use underscore character in the host name.

Web Server Configurations

This directory serves as a container for the configuration properties for all Web servers associated with the selected integration server. If all Web servers in the IS share common properties, these properties can be set in the *Default Web Server*. Otherwise, you should create separate Web Servers as described below.

To create a new set of Web server configuration properties

1. Right-click **Web Server Configurations** to display the context menu, and click **Create New Section**.
2. Give the newly created section an appropriate name.
3. In the properties for the newly created section, make any necessary changes to the default properties. The **port** number must be different for each Web Server.
4. Click **OK**.

Default Web Server

The Default Web Server properties serve as a template for the individual Web servers.

Figure 304 Default Web Server Properties

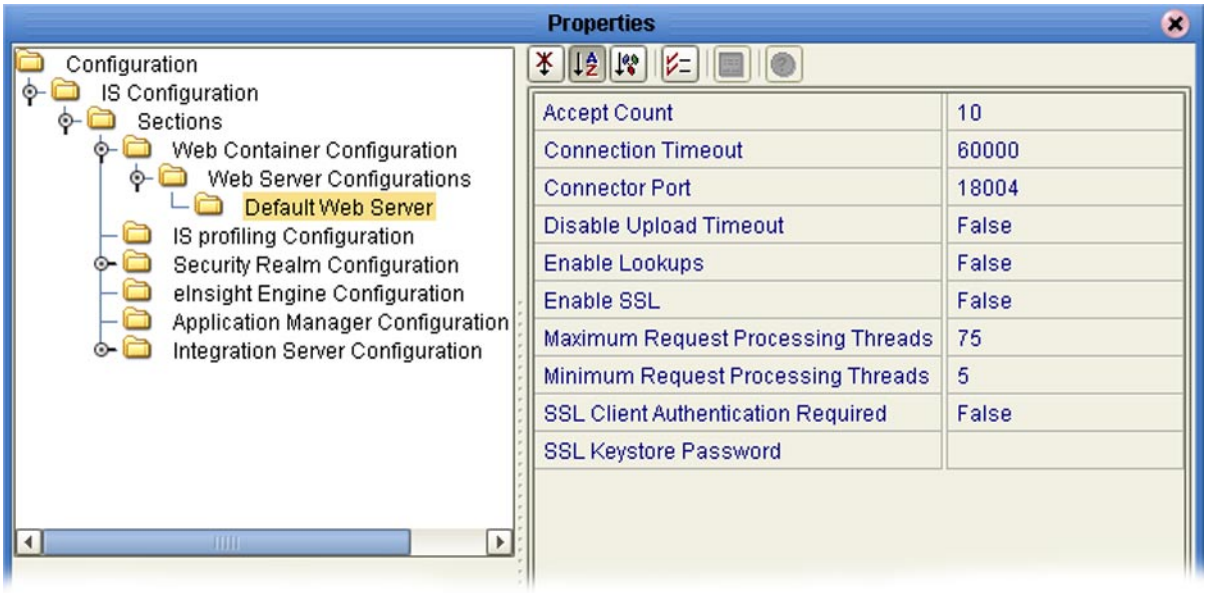


Table 101 Default Web Server Properties List

Property	Description
Accept Count	Specifies the maximum acceptable number of incoming connection requests when all possible request processing threads are in use. Any requests received beyond this number when the queue is full are refused. The default value is **10**.
Connection Timeout	Specifies the time period in milliseconds that this connector will wait for the request URI line to be presented, after accepting the connection. The default value is **60,000** ms (1 min).
Connector Port	Specifies the connection port for the Web server. The port numbers are dynamically assigned; the number shown is typical.
Disable Upload Timeout	Allows the servlet container to use a different, and longer, connection timeout while a servlet is executing. This gives the servlet a longer time to complete execution, and/or provides a longer timeout during data upload. The default value is **False**.
Enable Lookups	If set to **True**, calls are made requesting getRemoteHost() to perform DNS lookups in order to return the actual host name of the remote client. If set to **False**, the DNS lookup is bypassed and the IP address is returned in string form, thereby improving performance. The default value is **False**.

Table 101 Default Web Server Properties List

Property	Description
Enable SSL	Specifies whether or not to enable the Secure Sockets Layer (SSL) protocol. The default value is **False**.
Maximum Request Processing Threads	Specifies the maximum number of request processing threads to be created by this connector, thereby determining the maximum number of simultaneous requests that can be handled. The default value is **75**.
Minimum Request Processing Threads	Specifies the number of request processing threads to be created by this connector when it is first started. This value must be less than the value set for the Maximum Request Processing Threads property. The default value is **5**.
SSL Client Authentication Required	Set this property to **True** if you want to require SSL Client Authentication. The default value is **False**.
SSL Keystore Password	Enter the password you used when creating the certificate (there is no default value).

Note: *See the eGate Integrator System Administration Guide for additional information regarding SSL setup.*

Performance Monitoring (Profiling) Configuration

You can monitor the performance of the integration server by using the built-in *Heap Analysis* tool, which is enabled and configured using the Profiling Configuration dialog box (see Figure 305).

Figure 305 Profiling Configuration Properties

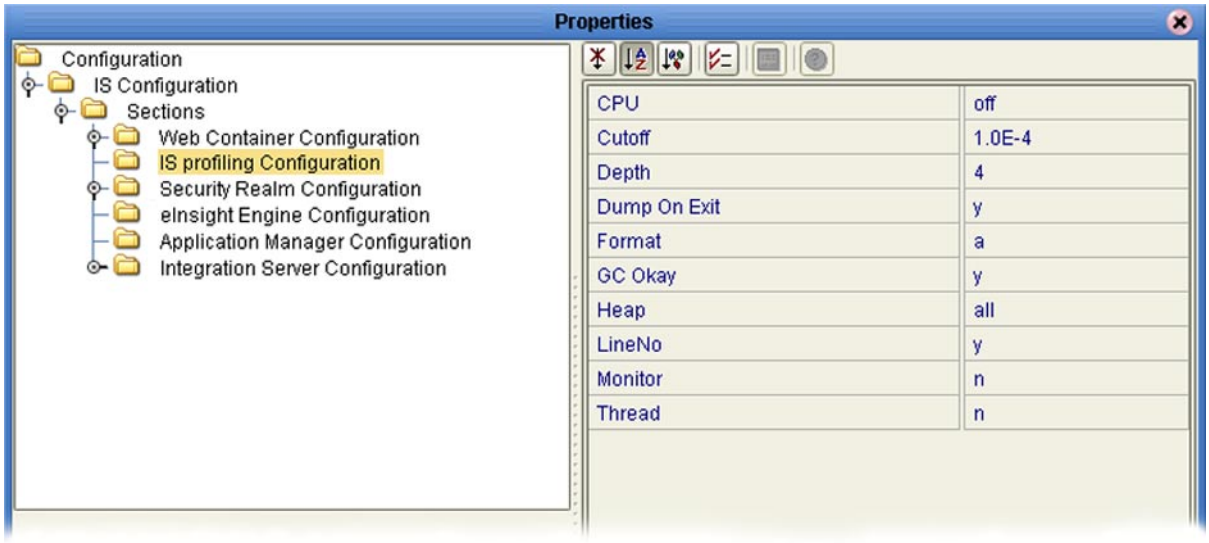


Table 102 Profiling Configuration Properties List

Property	Description
CPU	Specifies whether or not CPU usage is included in the trace. The default value is **off**.
Cutoff	Specifies the output cutoff point. The default value is **1.0E-4**.
Depth	Specifies the stack trace depth. The default value is **4**.
Dump on Exit	Specifies whether or not to dump on exit. The default value is **y** (yes).
Format	Specifies ASCII (**a**) or binary (**b**) output. The default value is **a** (ASCII).
GC Okay	Specified whether or not to allow garbage collection (GC) during sampling. The default value is **y** (yes).
Heap	Specifies the blocks of memory to include in traces. The default value is **all**.
LineNo	Specifies whether or not to include line numbers in traces. The default value is **y** (yes).
Monitor	Specifies whether or not to include monitor contention. The default value is **n** (no).
Thread	Specifies whether or not to include the thread in traces. The default value is **n** (no).

Security Realm Configuration

These properties pertain to the Lightweight Directory Access Protocol (LDAP), if used. Subdirectories contain properties for Sun Java System, Microsoft Active Directory Server, and OpenLDAP Directory Server. See the *eGate Integrator System Administration Guide* for information regarding Security Realm configuration.

Figure 306 Security Configuration Properties

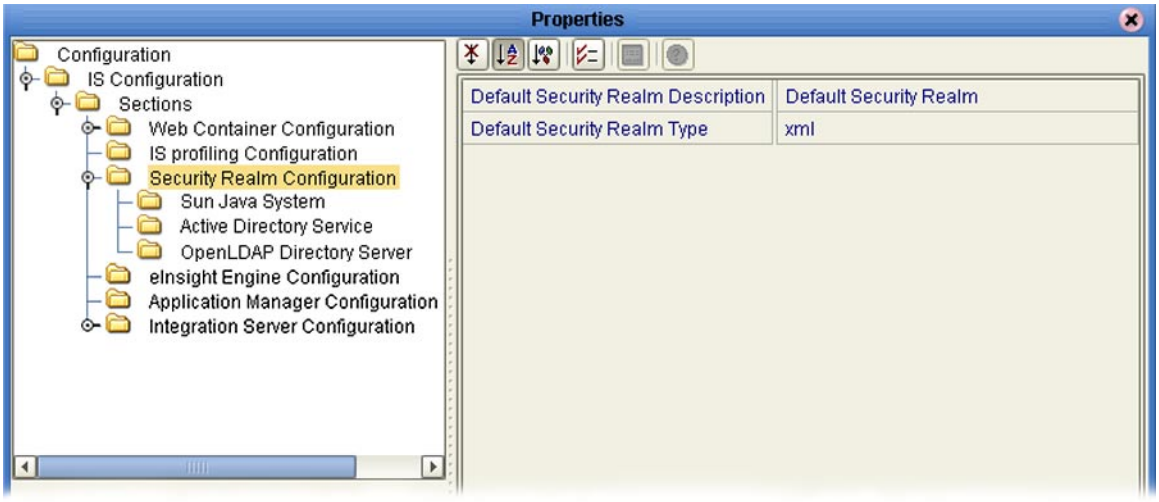


Table 103 Security Realm Configuration Properties List

Property	Description
Default Security Realm Description	Specifies the name for the default LDAP Security Realm. The default value is **Default Security Realm**.
Default Security Realm Type	Specifies the default LDAP Security Realm type. The default value is **xml**.

Security Realm Configuration: Sun Java System

These configuration properties are used only if you are using the Sun Java System (formerly SunONE) as an LDAP server. See the *eGate Integrator System Administration Guide* for additional information.

Figure 307 Sun Java System Configuration Properties

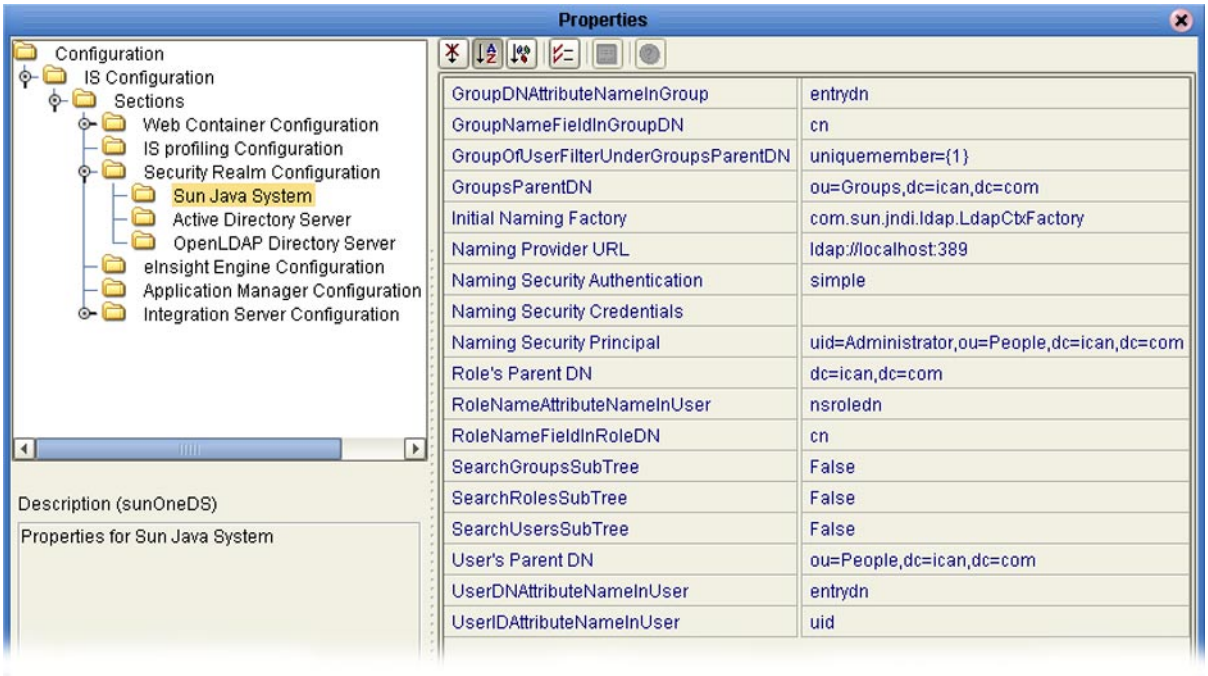


Table 104 Sun Java System Configuration Properties List

Property	Default Value
GroupDNAttributeNameInGroup	entrydn
GroupNameFieldInGroupDN	cn
GroupOfUserFilterUnderGroupsParentDN	uniquemember={1}
GroupsParentDN	cn=Groups,dc=ican,dc=com
Initial Naming Factory	com.sun.jndi.ldap.LdapCtxFactory
Naming Provider URL	ldap://localhost:389
Naming Security Authentication	simple
Naming Security Credentials	*(password)*
Naming Security Principle	uid=Administrator,ou=People,dc=ican,dc=com
Role's ParentDN	dc=ican,dc=com
RoleNameAttributeNameInUser	nsroledn
RoleNameFieldInRoleDN	cn
SearchGroupsSubTree	False
SearchRolesSubTree	False

Table 104 Sun Java System Configuration Properties List

Property	Default Value
SearchUsersSubTree	False
User's ParentDN	cn=People,dc=ican,dc=com
UserDNAttributeNameInUser	entrydn
UserIDAttributeNameInUser	uid

Security Realm Configuration: Active Directory Service

These configuration properties are used only if you are using the Microsoft Active Directory Server as an LDAP server. See the *eGate Integrator System Administration Guide* for additional information.

Figure 308 Active Directory Server Configuration Properties

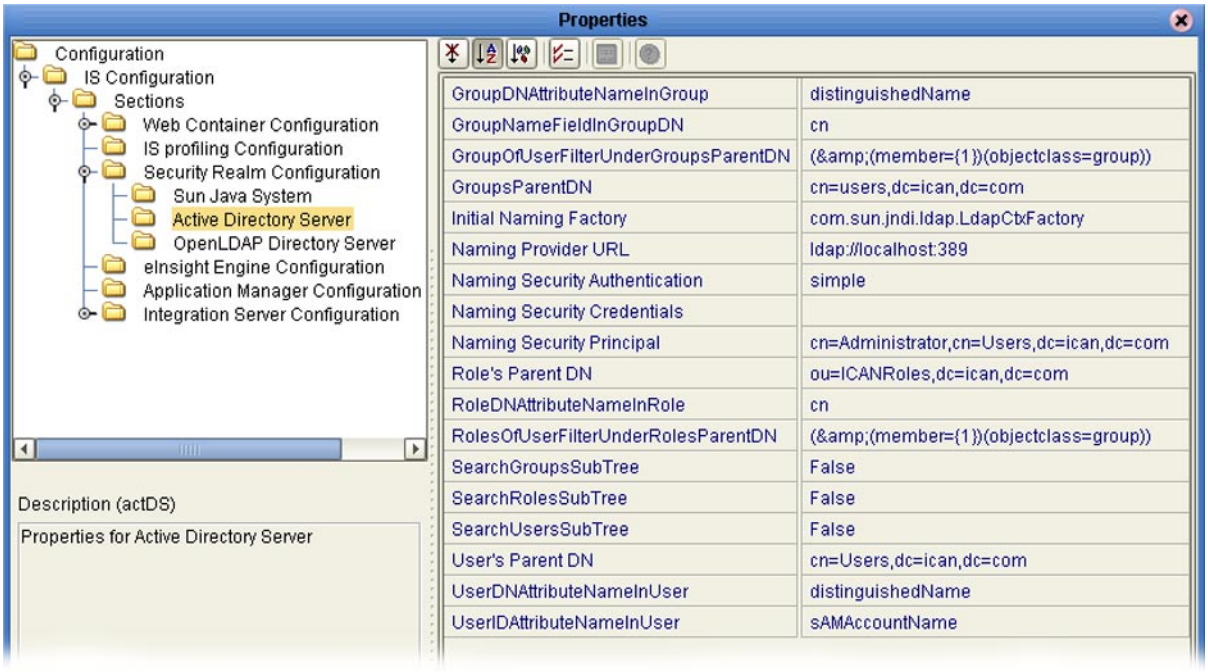


Table 105 Active Directory Server Configuration Default Properties List

Property	Default Value
GroupDNAttributeNameInGroup	distinguishedName
GroupNameFieldInGroupDN	cn
GroupOfUserFilterUnderGroupsParentDN	(&(member={1})(objectclass=group))
GroupsParentDN	cn=users,dc=ican,dc=com
Initial Naming Factory	com.sun.jndi.ldap.LdapCtxFactory
Naming Provider URL	ldap://localhost:389
Naming Security Authentication	simple
Naming Security Credentials	*(password)*
Naming Security Principle	cn=Administrator,cn=Users,dc=ican,dc=com
Role's Parent DN	ou=ICANRoles,dc=ican,dc=com
RoleDNAttributeNameInRole	cn
RolesOfUserFilterUnderGroupsParentDN	(&(member={1})(objectclass=group))
SearchGroupsSubTree	False
SearchRolesSubTree	False

Table 105 Active Directory Server Configuration Default Properties List

Property	Default Value
SearchUsersSubTree	False
User's Parent DN	cn=Users,dc=ican,dc=com
UserDNAttributeNameInUser	distinguishedName
UserIDAttributeNameInUser	sAMAccountName

Security Realm Configuration: OpenLDAP Directory Server

These configuration properties are used only if you are using the OpenLDAP Directory Server as an LDAP server. See the *eGate Integrator System Administration Guide* for additional information.

Figure 309 OpenLDAP Directory Server Configuration Properties

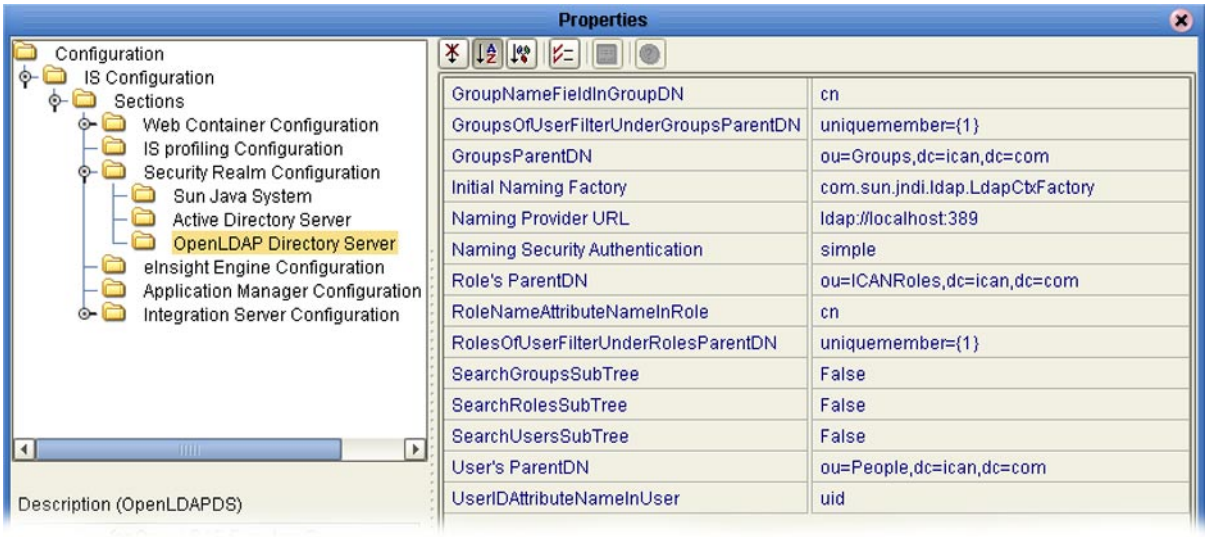


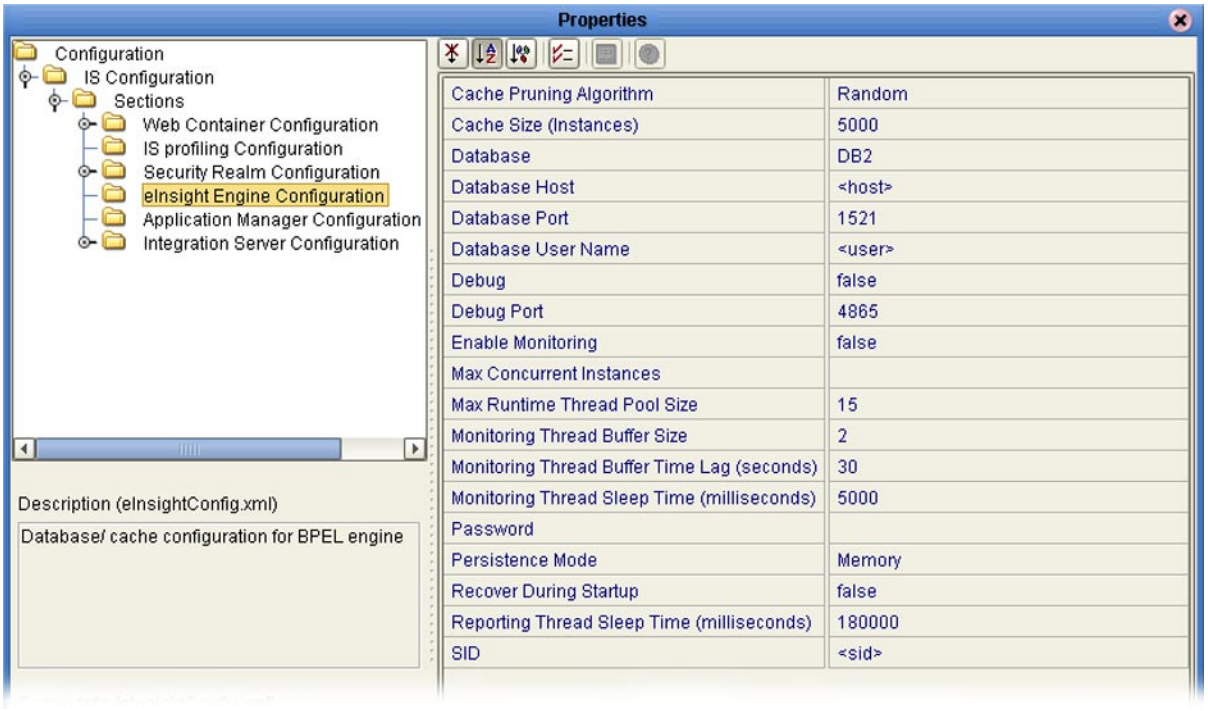
Table 106 OpenLDAP Directory Server Configuration Default Properties List

Property	Default Value
GroupNameFieldInGroupDN	cn
GroupOfUserFilterUnderGroupsParentDN	uniquemember={1}
GroupsParentDN	ou=Groups,dc=ican,dc=stc.com
Initial Naming Factory	com.sun.jndi.ldap.LdapCtxFactory
Naming Provider URL	ldap://localhost:389
Naming Security Authentication	simple
Role's ParentDN	ou=ICANRoles,dc=ican,dc=stc.com
RoleNameAttributeNameInRole	cn
RolesOfUserFilterUnderGroupsParentDN	uniquemember={1}
SearchGroupsSubTree	False
SearchRolesSubTree	False
SearchUsersSubTree	False
User's ParentDN	cn=People,dc=ican,dc=com
UserIDAttributeNameInUser	uid

eInsight Engine Configuration

This configuration node is operational only if you have eInsight Business Process Manager installed on your system. The configuration properties relate to the BPEL engine's database cache; see the *eInsight Business Process Manager User's Guide* for information regarding these properties (see Figure 310).

Figure 310 eInsight Engine Configuration Properties



Application Manager Configuration

You can set integration server thread pool variables using the Application Manager Configuration Properties dialog box (see Figure 311).

Figure 311 Application Manager Configuration Properties

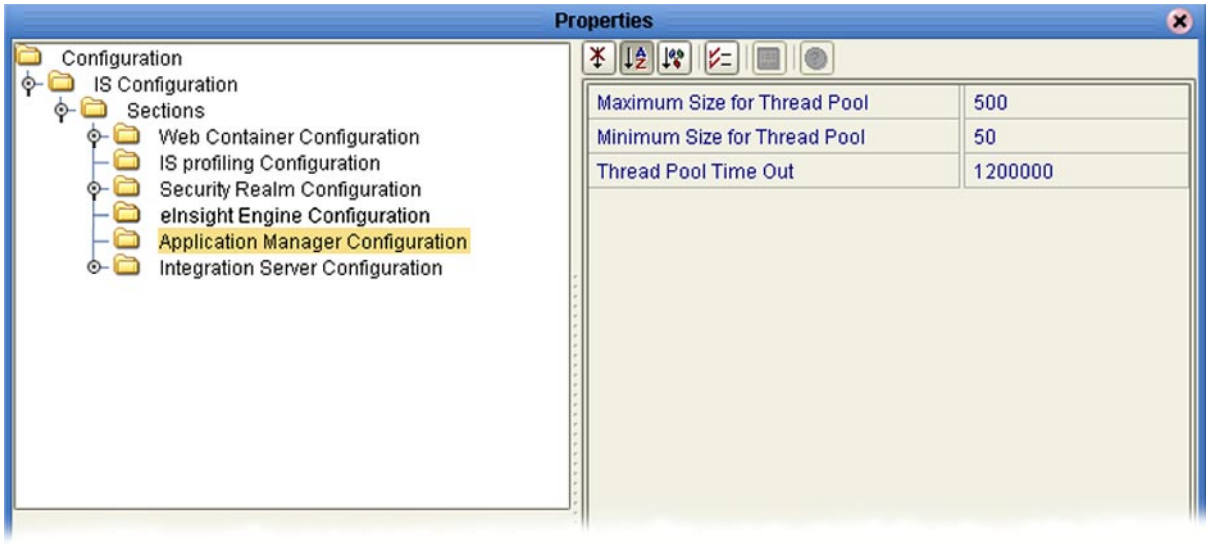


Table 107 Application Manager Configuration Properties List

Property	Description
Maximum Size for Thread Pool	Specifies the maximum number of threads that can exist in the thread pool. The default value is **500**.
Minimum Size for Thread Pool	Specifies the minimum number of threads for the thread pool. The default value is **50**.
Thread Pool Time Out	Specifies the timeout interval for the thread pool, measured in milliseconds. The default value is **1,200,000** ms (20 min).

Integration Server Configuration

Detailed, low-level configuration of the integration server is performed using the Integration Server Configuration Properties dialog box (see Figure 312).

Figure 312 Integration Server Configuration Properties

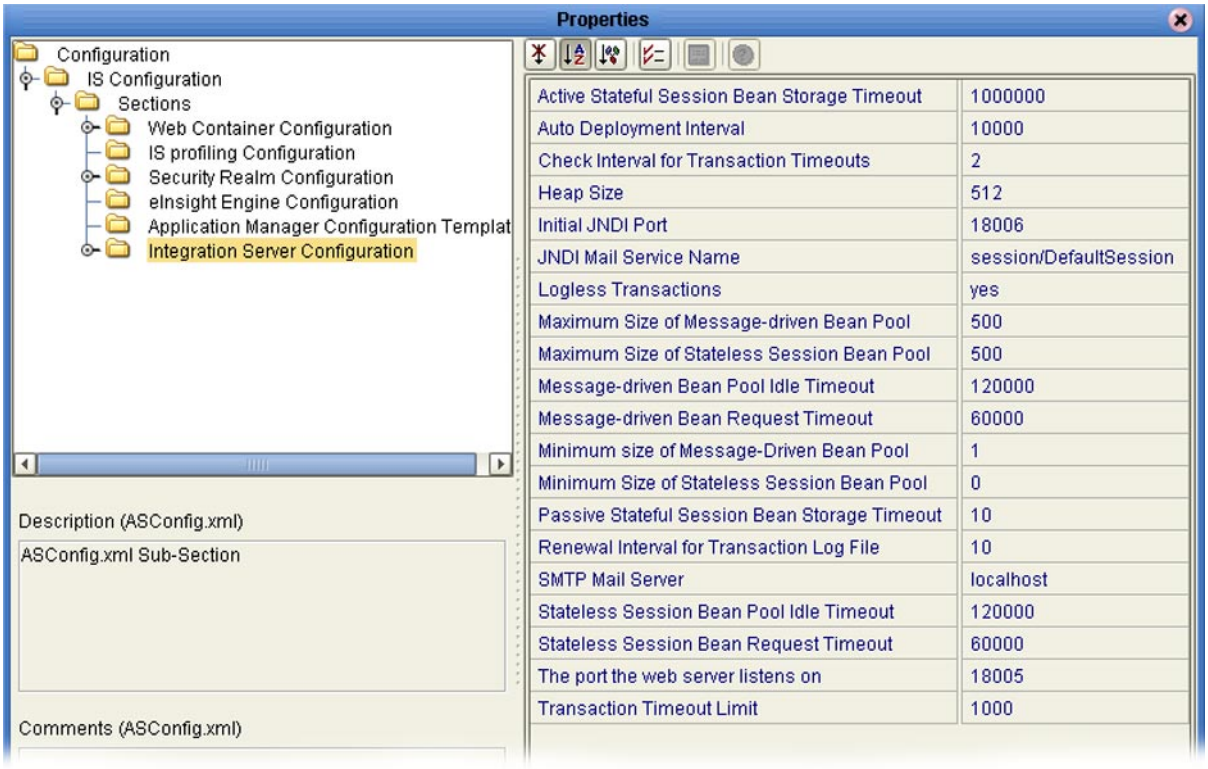


Table 108 Integration Server Configuration Properties List

Property	Description
Active Stateful Session Bean Storage Timeout	Specifies the interval after which an Active Stateful Session Bean is removed from storage, measured in minutes. The default value is **1,000,000** min, which ensures that it will not be removed unintentionally.
Auto Deployment Interval	Specifies the interval at which the auto-deployer checks the deployment directory for files, measured in milliseconds. The default value is **10,000** ms.
Check Interval for Transaction Timeouts	Specifies the interval between checks for transaction timeouts, measured in minutes. The default value is **2** min.
Heap Size	Specifies the Heap size in Megabytes; the minimum size is **512** Mb, which is the default value. Increasing this value increases the JVM size.
Initial JNDI Port	Specifies the initial port required by the Naming Service class for startup. The port numbers are dynamically assigned; the number shown is typical.

Table 108 Integration Server Configuration Properties List

Property	Description
JNDI Mail Service Name	Specifies the name of the Java Naming and Directory Interface (JNDI) mail service. The default value is **session/DefaultSession**.
Logless Transactions	Specifies whether or not logless transactions are allowed. The default value is **yes**.
Maximum Size of Message-driven Bean Pool	Specifies the maximum number of Message-driven Beans allowed in the Message-driven Bean pool at one time. The default value is **500**.
Maximum Size of Stateless Session Bean Pool	Specifies the maximum number of Stateless Session Beans allowed in the Stateless Session Bean pool at one time. The default value is **500**.
Message-driven Bean Pool Idle Timeout	Specifies the timeout interval for the Message-driven Bean pool, measured in milliseconds. The default value is **120,000** ms (2 min).
Message-driven Bean Request Timeout	Specifies the interval after which a Message-driven Bean request times out, measured in milliseconds. The default value is **60,000** ms (1 min).
Minimum Size of Message-driven Bean Pool	Specifies the minimum number of Message-driven Beans allowed in the Message-driven Bean pool at one time. The default value is **1**.
Minimum Size of Stateless Session Bean Pool	Specifies the maximum number of Stateless Session Beans allowed in the Stateless Session Bean pool at one time. The default value is **1**.
Passive Stateful Session Bean Storage Timeout	Specifies the interval after which a Passive Stateful Session Bean is removed from storage, measured in minutes. The default value is **10** min.
Renewal Interval for Transaction Log File	Specifies the interval for renewing the Transaction Service log file, measured in hours. The default value is **10** hr.
SMTP Mail Server	Specifies the name of the SMTP mail host server. The default value is **localhost**. Note: Do not use underscore character in the host name.
Stateless Session Bean Pool Idle Timeout	Specifies the timeout interval for the Stateless Bean pool, measured in milliseconds. The default value is **120,000** ms (2 min).
Stateless Session Bean Pool Request Timeout	Specifies the interval after which a Stateless Bean request times out, measured in milliseconds. The default value is **60,000** ms (1 min).
The Port the Web Server Listens On	Specifies the port the Web server listens on. The port numbers are dynamically assigned; the number shown is typical.
Transaction Timeout Limit	Specifies the time limit for transactions to time out, measured in seconds. The default value is **1,000** sec.

JDBC DataSource Connection Pools

This directory serves as a container for the configuration properties for all JDBC DataSource Connection Pools associated with the selected integration server. The only set of default configuration properties currently furnished is for Oracle JDBC.

Oracle JDBC Connection Pool

Connection Pool properties for an Oracle database associated with the integration server are specified in the Oracle JDBC Connection Pool dialog box (see Figure 313).

Figure 313 Oracle JDBC Connection Pool Properties

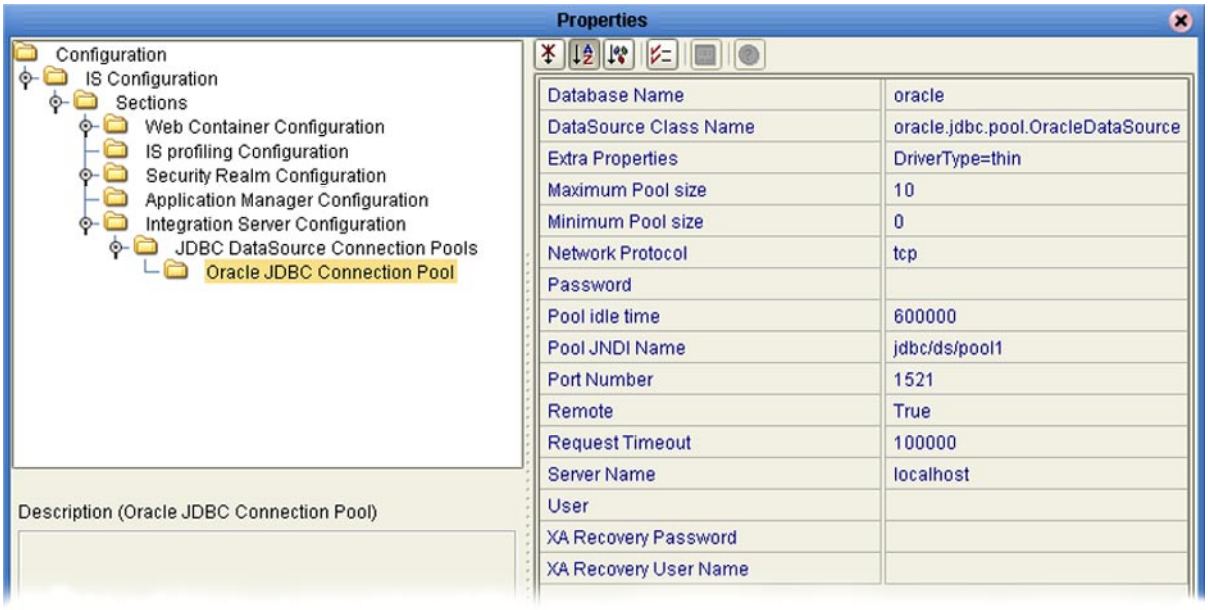


Table 109 Oracle JDBC Connection Pool Properties List

Property	Description
Database Name	Specifies the name of the database for which the pool is created. The default value is **oracle**.
DataSource Class Name	Specifies the name of the DataSource class. The default value is **oracle jdbc pool OracleDataSource**.
Extra Properties	Specifies custom properties for the DataSource, using semicolon-separated key-value pairs. The default value is **DriverType=thin**.
Maximum Pool Size	Specifies the maximum number of connections in the pool. The default value is **10**.
Minimum Pool Size	Specifies the minimum number of connections in the pool. The default value is **0**.
Network Protocol	Specifies the network protocol. The default value is **tcp**.

Table 109 Oracle JDBC Connection Pool Properties List

Property	Description
Password	Specifies the password for the connection. There is no default value.
Pool Idle Time	Specifies the maximum time period in milliseconds that a connection may remain unused before it is removed from the pool in order to reduce the pool size. The default value is **600,000** ms (10 min).
Pool JNDI Name	Specifies the unique Java Naming and Directory Interface (JNDI) name of the DataSource pool. The pool is bound in the java/namespace for local access or into the global namespace for remote access. The default value is **jdbc/ds/pool1**.
Port Number	Specifies the port number on which the server receives data. The default value is **1521**.
Remote	Specifies whether or not the DataSource should be bound into the global remote JINI namespace for access by remote clients. The default value is **true**.
Request Timeout	Specifies the maximum time period in milliseconds that a request for connection from the pool may block all other connections currently in use. The default value is **100,000** ms.
Server Name	Specifies the host name of the database server or IP address where the database server is running. The default value is **localhost**. Note: Do not use underscore character in the host name.
User	Specifies the user name authorized for creating connections. There is no default value.
XA Recovery Password	For XA DataSources only, specifies the password to use for XA transaction recovery. There is no default value.
XA Recovery User Name	For XA DataSources only, specifies the user name to use for XA transaction recovery. There is no default value.

9.5.2 Deploying User-Defined Stateless Session Beans

User-defined stateless session beans can be deployed to the SeeBeyond Integration Server following the procedure outlined in this section.

Note: *The deployment of stateful session beans, entity beans, and message-driven beans is not currently supported.*

To deploy a stand-alone SLSB to the SeeBeyond Integration Server

1. Create and compile the EJB.
2. Write the **ejb-jar.xml** and **seebeyond-ejb.xml** deployment descriptors for your EJB.
3. Create a **.jar** file with the deployment descriptors in the **\META-INF** directory and the code in the root.
4. Move the **.jar** file into the **\logicalhost\stcis\deploy\new***integration_server_name* directory for deployment. The Integration Server will automatically pick up the **.jar** file from this location and deploy the EJB.

Examples of the EJBs and associated **.xml** files are as follows.

Example Remote Interface

```
package ejb.CustomApp;
import java.rmi.RemoteException;
import java.rmi.Remote;
import javax.ejb.*;

public interface CustomApp extends EJBObject, Remote
{
  public String getId() throws RemoteException;
}
```

Example Home Interface

```
package ejb.CustomApp;

import javax.ejb.*;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.*;

public interface CustomAppHome extends EJBHome
{
   public CustomApp create() throws CreateException, RemoteException;
}
```

Example Stateless Session Bean (SLSB)

```
package ejb.CustomApp;

import javax.ejb.*;
import java.io.Serializable;
import java.util.*;
import java.rmi.*;
import javax.naming.Context;
import javax.naming.InitialContext;

// import addtional classes as needed "CustomController"
```

```
public class CustomAppBean implements SessionBean
{
  private SessionContext ctx;
  private CustomController mCustom;

  public void setSessionContext( SessionContext context )
  {
     this.ctx = context;
  }

  public void ejbCreate()
  {
     try {
       javax.naming.Context context = new InitialContext();
       // lookup Custom application
       Object ref = context.lookup("ejb/CustomController");
       CustomControllerHome CustomHome =
(CustomControllerHome)javax.rmi.PortableRemoteObject.narrow(ref,
CustomControllerHome.class);
       mCustom = CustomHome.create();
     } catch (Exception e) {
     System.out.println( e.getMessage() );
     }
  }

  public String getId()
  {
     SystemObjectPK key = new SystemObjectPK( "SBYN", "0000000001" );
     String EUID= "Not Found";
     try {
    EUID = mCustom.getEUID( key );
     }
     catch (Exception e) {
     System.out.println("===> Exception: " );
     System.out.println( e.getMessage() );
     }

     return( EUID );
  }

  // add addtional EJB methods

}
```

Example ejb-jar.xml file for the above SLSB

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise
JavaBeans 2.0//EN' 'http://java.sun.com/dtd/ejb-jar_2_0.dtd'>
<!-- Generated XML! -->
<ejb-jar>
    <display-name>ServiceBeans</display-name>
    <enterprise-beans>
        <session>
            <description><![CDATA[Custom App Session Bean]]></
description>
            <display-name>Custom App</display-name>
            <ejb-name>CustomApp</ejb-name>
            <home>ejb.CustomApp.CustomAppHome</home>
            <remote>ejb.CustomApp.CustomApp</remote>
            <ejb-class>ejb.CustomApp.CustomAppBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Bean</transaction-type>
            <ejb-ref>
                <ejb-ref-name>ejb/CustomController</ejb-ref-name>
```

```
                    <ejb-ref-type>Session</ejb-ref-type>

<home>com.stc.eindex.ejb.Custom.CustomControllerHome</home>
                    <remote>com.stc.eindex.ejb.Custom.CustomController</
remote>
                    <ejb-link>CustomController</ejb-link>
                </ejb-ref>
            </session>
        </enterprise-beans>
</ejb-jar>
```

Example seebeyond-ejb.xml file for the above SLSB

```
<sbyn-ejb-deployment-descriptor>
    <enterprise-beans>
        <session>
            <ejb-name>CustomApp</ejb-name>
            <jndi-name>ejb/CustomApp</jndi-name>
            <security>
                <authorize>no</authorize>
                <authenticate>no</authenticate>
                <security-audit>no</security-audit>
            </security>
            <pool-min>1</pool-min>
        </session>
    </enterprise-beans>
</sbyn-ejb-deployment-descriptor>
```

SLSB Deployment verification

Examine the log file **\logicalhost\logs\stc_is_*integration_server_name*.log**. You should find text such as "**CustomApp (EJB) was successfully deployed**" confirming deployment.

To remove a stand-alone SLSB from the SeeBeyond Integration Server

1. Shut down the Logical Host containing the Integration Server where the SLSB is deployed.
2. Remove the **.jar** file created in the preceding deployment procedure from the **\logicalhost\stcis\repository\applications*integration_server_name*\EJB** directory.
3. Restart the Logical Host.

9.6 Message Servers

The Logical Host contains one or more Message Servers, which manage JMS topics (publish-and-subscribe messaging) and queues (point-to-point messaging). eGate Integrator includes the SeeBeyond JMS IQ Manager as its Java Messaging Service (JMS) implementation. The JMS IQ Manager conforms to the Java Message specification 1.0.2b, and supports both topic (publish-and-subscribe) and queue (point-to-point) messaging styles.

eGate Integrator also includes support for HP NonStop JMS for eGate Integrator implementations on HP NonStop Server platforms. Third-party application servers such as BEA WebLogic and IBM WebSphere incorporate their own message servers.

This section presents an overview of the configuration properties for the SeeBeyond JMS IQ Manager. For more information on the JMS IQ Manager, and deploying Project components to third-party message servers, see the *eGate Integrator JMS Reference Guide*.

9.6.1 SeeBeyond JMS IQ Manager Configuration

General Configuration

These properties cover the basic configuration of the JMS IQ Manager.

Figure 314 JMS IQ Manager - General Configuration Properties

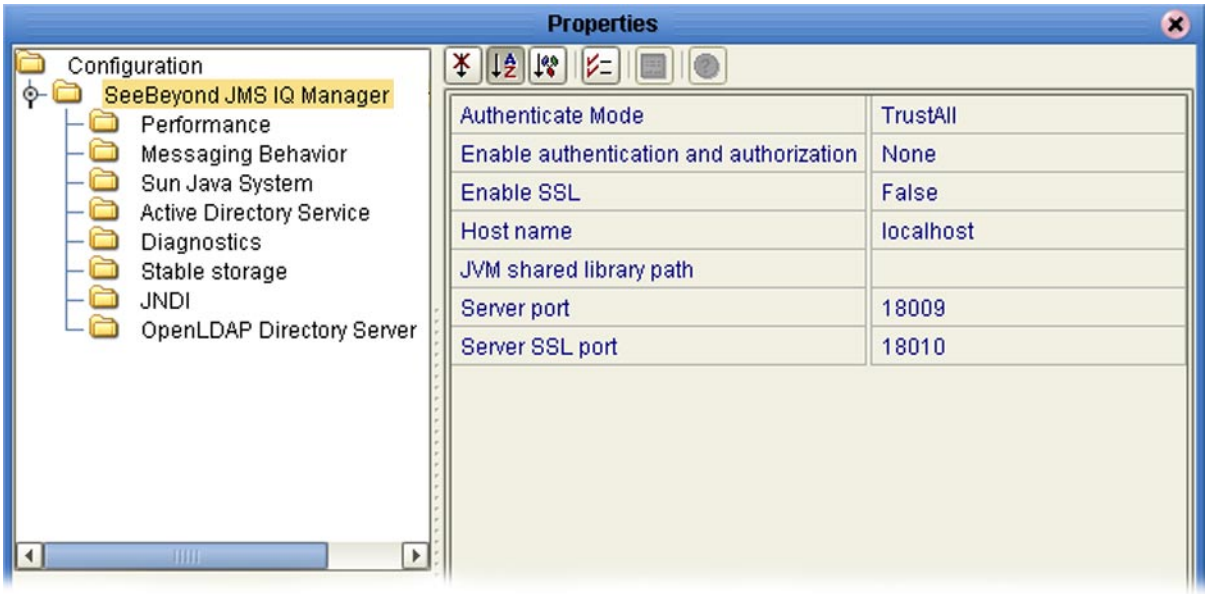


Table 110 JMS IQ Manager - General Configuration Properties List

Property	Description
Authenticate Mode	Specifies whether or not the JMS IQ Manager authenticates components attempting to connect to it. The options are **Authenticate** and **TrustAll**; the default value is **TrustAll**.
Enable authentication and authorization	Specifies whether or not security is enabled for the JMS IQ Manager, requiring a user name and password. The options are: ▪ **None** (no security—default value) ▪ **File** (use File realm) ▪ **SunONE** (use SunONE LDAP server) ▪ **AD** (use Active Directory LDAP server) If you specify either **SunONE** or **AD**, you must also configure your LDAP server. See **Sun Java System** on page 386 and **Active Directory Service** on page 388.
Enable SSL	Specifies whether or not to enable the Secure Sockets Layer (SSL) protocol for the JMS IQ Manager's TCP/IP connections. The options are **True** and **False**; the default value is **False**.
Host name	Specifies the name of the host system for the JMS IQ Manager. The default value is **localhost**. Note: Do not use underscore character in the host name.
JVM shared library path	Specifies the path to the Java Virtual Machine shared library. The value depends on the operating system under which the Logical Host is running—use the ellipsis (…) button to select the correct value. The default value is **../../jre/bin/client/jvm.dll**, for Windows platforms.
Server port	Specifies the TCP/IP port number. The port numbers are dynamically assigned; the number shown is typical. Each JMS IQ Manager must have a unique port number per system.
Server SSL port	Specifies the port on which the JMS IQ Manager listens for SSL connections. The port numbers are dynamically assigned; the number shown is typical.

Performance

The Performance properties allow you to specify memory usage for optimum system performance. See the *eGate Integrator JMS Reference Guide* for additional information.

Figure 315 Performance Configuration Properties

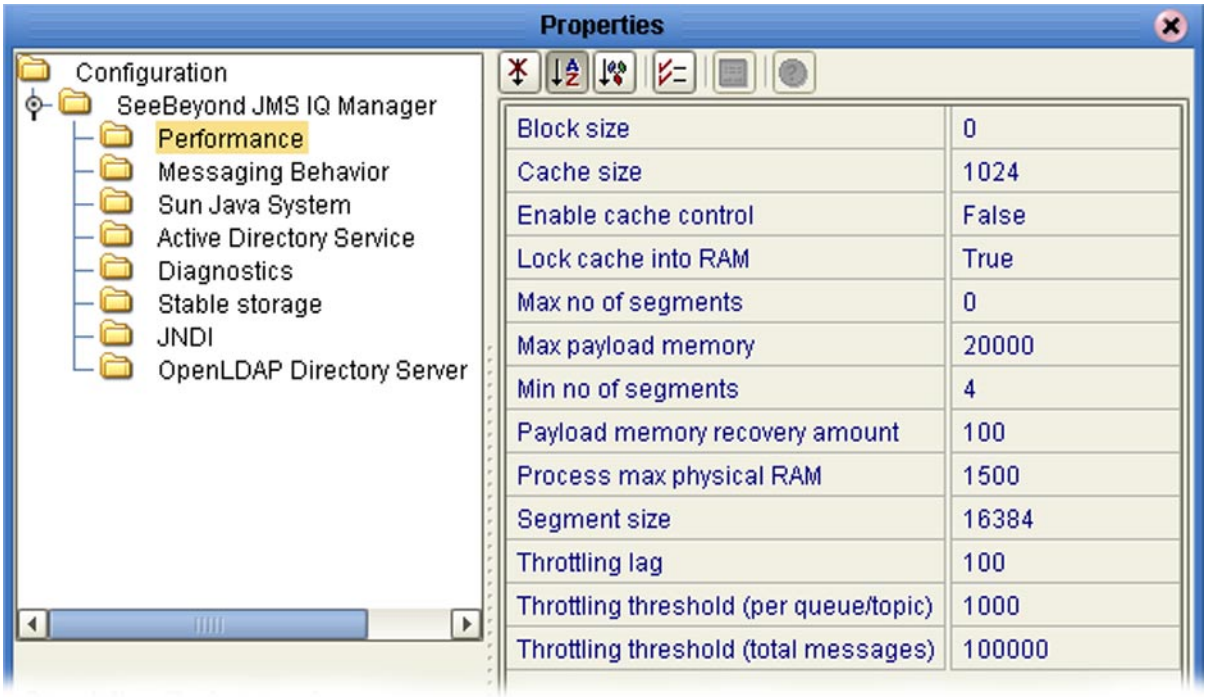


Table 111 Performance Configuration Properties List

Property	Description
Block Size	Specifies the block size used for DBS files. Allowed values are **0**, **512**, and **1024**. If you specify **0**, the server will automatically determine the value to use by querying the operating system. **WARNING**: If you change this value, your existing DBS files will be unusable with the new setting.
Cache size	Specifies the total number of pages in the *read* cache (a page is 512 bytes on Windows and 1024 bytes on UNIX). The default value is **1024** pages (0.5 MB for Windows, 1 MB for UNIX).
Enable cache control	Specifies whether or not the JMS IQ Manager controls the cache synchronization to disk. The options are **True** and **False**; the default is **False**.
Lock cache into RAM	Specifies whether or not the cache is locked into physical memory. The options are **True** and **False**; the default is **True**.

Table 111 Performance Configuration Properties List

Property	Description
Max. no. of segments	Specifies the upper limit for the number of database files that the JMS IQ Manager creates for its stable message storage. The default value is **0**, which causes the JMS IQ Manager to create files as needed—limited only by disk space.
Max. payload memory	Specifies the upper limit for the amount of cache, in kB, allocated for the server to keep message payloads in cache. The default value is **20,000** kB.
Min. no. of segments	Specifies the minimum number of database files that the JMS IQ Manager initially creates for its stable message storage. The default value is **4** segments (files).
Payload memory recovery amount	Specifies the amount of cache, in kB, to recover in a recovery and cleanup operation. The default value is **100** kB.
Process max. physical RAM	Specifies the upper limit for the amount of RAM, in kB, allocated for use by the JMS IQ Manager as working memory (Windows platforms only). The default value is **1500** kB.
Segment size	Specifies the total number of pages in a single database file (segment). The default value is **16,384** pages (8 MB for Windows, 16 MB for UNIX).
Throttling lag	Specifies the number of messages that must be dequeued before message producers are no longer throttled. The default value is **100**.
Throttling threshold (per queue/topic)	Specifies the maximum number of messages after which all message producers for the message destination (topic or queue) are throttled. The default value is **1000** messages.
Throttling threshold (total messages)	Specifies the maximum number of messages for all message destinations, after which the JMS IQ Manager begins throttling producers. The default value is **100,000** messages.

Messaging Behavior

The Messaging Behavior properties allow you to configure the order of message delivery. See the *eGate Integrator JMS Reference Guide* for additional information.

Figure 316 Messaging Behavior Configuration Properties

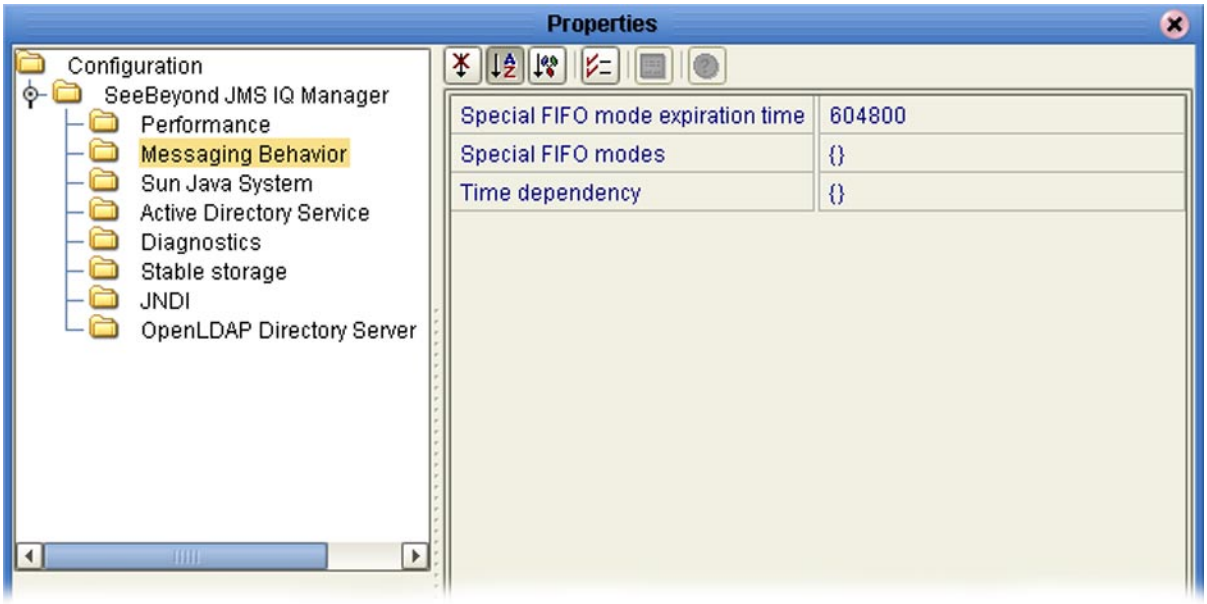


Table 112 Messaging Behavior Configuration Properties List

Property	Description
Special FIFO mode expiration time	Specifies the expiration time in seconds for first-in, first-out (FIFO) mode operation. The default value is **604,800** seconds (7 days).
Special FIFO modes	Specifies the FIFO delivery order: ▪ fully concurrent (**0**) ▪ protected concurrent (**1**) ▪ fully serialized (**2**) The default value is (**0**). See the *eGate Integrator JMS Reference Guide* for specification syntax.
Time dependency	Specifies whether or not the processing order is dependent on messages that are associated with other destinations. The default value is **0**, indicating no such dependency. See the *eGate Integrator JMS Reference Guide* for syntax.

Sun Java System

These configuration properties are used only if you are using the Sun Java System (formerly SunONE) as an LDAP server. See the *eGate Integrator System Administration Guide* for additional information.

Figure 317 Sun Java System Configuration Properties

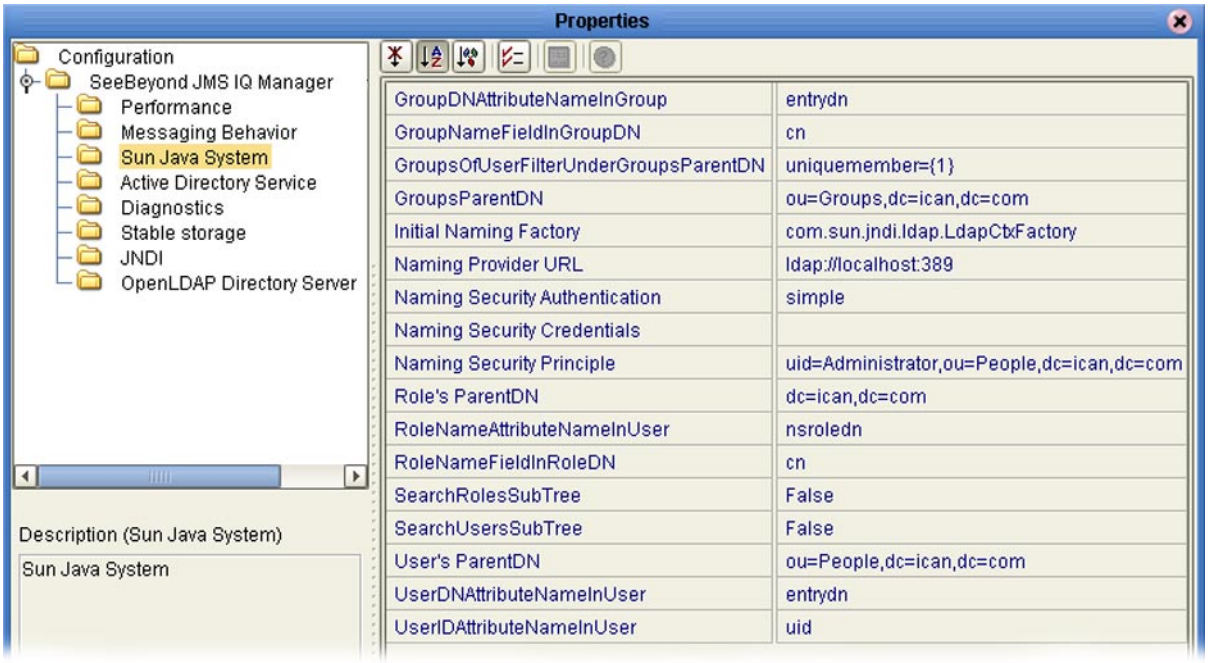


Table 113 Sun Java System Configuration Default Properties List

Property	Default Value
GroupDNAttributeNameInGroup	entrydn
GroupNameFieldInGroupDN	cn
GroupOfUserFilterUnderGroupsParentDN	uniquemember={1}
GroupsParentDN	cn=Groups,dc=ican,dc=com
Initial Naming Factory	com.sun.jndi.ldap.LdapCtxFactory
Naming Provider URL	ldap://localhost:389
Naming Security Authentication	none
Naming Security Credentials	*(password)*
Naming Security Principle	cn=Administrator,cn=Users,dc=ican,dc=com
Role's ParentDN	dc=ican,dc=com
RoleDNAttributeNameInRole	nsroledn
RolesOfUserFilterUnderGroupsParentDN	cn
SearchRolesSubTree	False

Table 113 Sun Java System Configuration Default Properties List

Property	Default Value
SearchUsersSubTree	False
User's ParentDN	cn=People,dc=ican,dc=com
UserDNAttributeNameInUser	entrydn
UserIDAttributeNameInUser	uid

Active Directory Service

These configuration properties are used only if you are using the Microsoft Active Directory Service as an LDAP server. See the *eGate Integrator System Administration Guide* for additional information.

Figure 318 Active Directory Service Configuration Properties

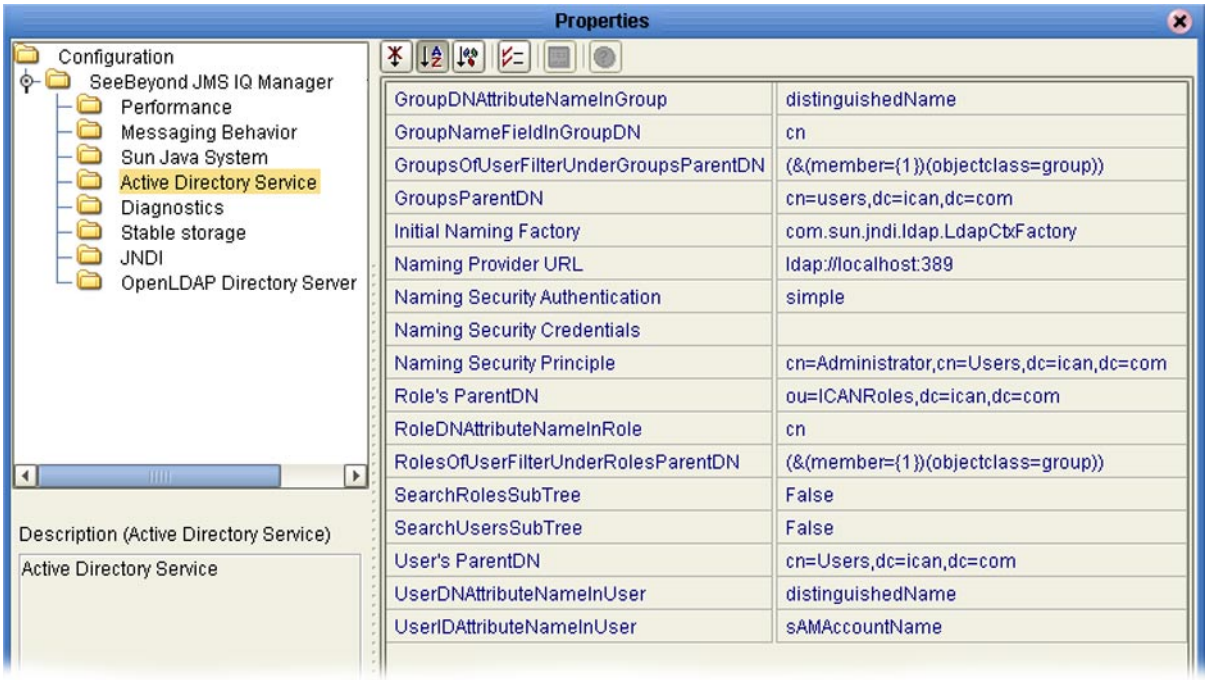


Table 114 Active Directory Service Configuration Default Properties List

Property	Default Value
GroupDNAttributeNameInGroup	distinguishedName
GroupNameFieldInGroupDN	cn
GroupOfUserFilterUnderGroupsParentDN	(&(member={1})(objectclass=group))
GroupsParentDN	cn=users,dc=ican,dc=com
Initial Naming Factory	com.sun.jndi.ldap.LdapCtxFactory
Naming Provider URL	ldap://localhost:389
Naming Security Authentication	simple
Naming Security Credentials	*(password)*
Naming Security Principle	cn=Administrator,cn=Users,dc=ican,dc=com
Role's ParentDN	ou=ICANRoles,dc=ican,dc=com
RoleDNAttributeNameInRole	cn
RolesOfUserFilterUnderGroupsParentDN	(&(member={1})(objectclass=group))
SearchRolesSubTree	False

Table 114 Active Directory Service Configuration Default Properties List

Property	Default Value
SearchUsersSubTree	False
User's ParentDN	cn=Users,dc=ican,dc=com
UserDNAttributeNameInUser	distinguishedName
UserIDAttributeNameInUser	sAMAccountName

Diagnostics

The Diagnostics properties allow you to configure the logging operations for the JMS IQ Manager. See the *eGate Integrator JMS Reference Guide* for additional information.

Figure 319 Diagnostics Configuration Properties

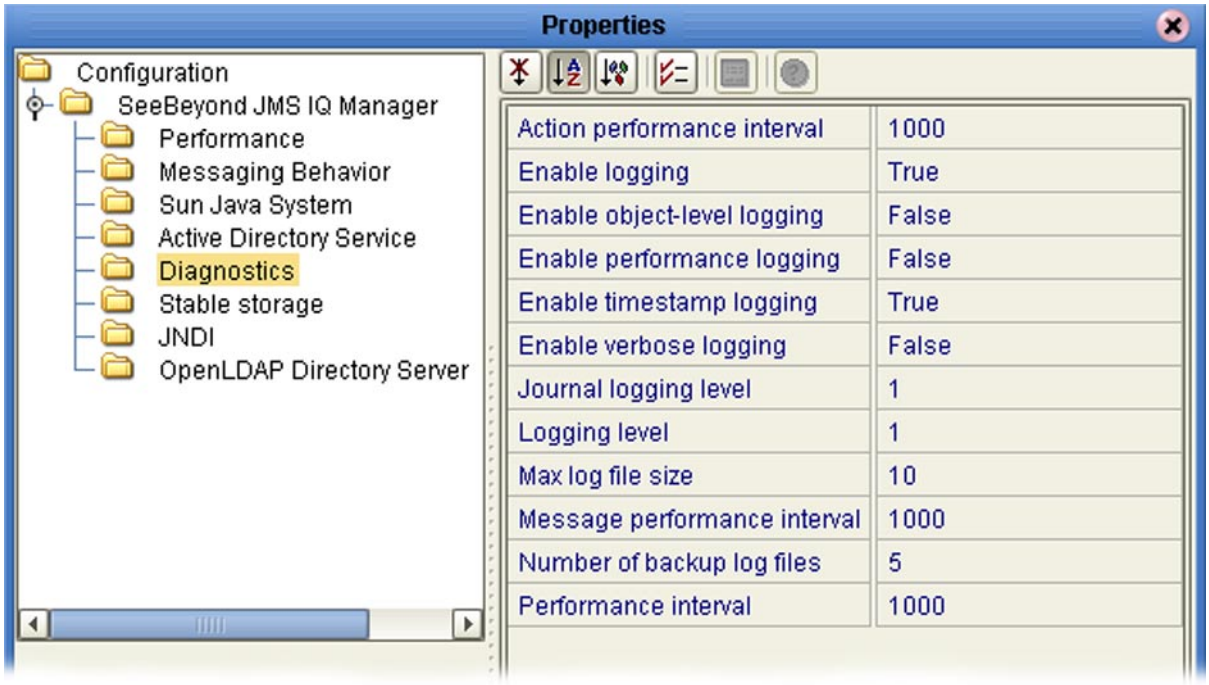


Table 115 Diagnostics Configuration Properties List

Property	Description
Action performance interval	Specifies how many times the ActionManager's *Update* function is called before logging its performance statistics. The default is **1000**.
Enable logging	Specifies whether or not diagnostic information is written to the JMS IQ Manager log file. **True** (the default) enables logging, **False** disables it.
Enable object-level logging	Specifies whether or not object (constructor/deconstructor) information is written to the JMS IQ Manager log file. **True** enables logging, **False** (the default) disables it.
Enable performance logging	Specifies whether or not performance information is written to the JMS IQ Manager log file. **True** enables logging, **False** (the default) disables it.

Table 115 Diagnostics Configuration Properties List

Property	Description
Enable timestamp logging	Specifies whether or not timestamp information is written to the JMS IQ Manager log file. **True** (the default) enables logging, **False** disables it.
Enable verbose logging	Specifies whether or not full-length messages are written to the JMS IQ Manager log file. **True** enables logging, **False** (the default) disables it.
Journal logging level	Specifies the threshold severity level at which the system issues informational, warning, and error messages, and writes them to the JMS IQ Manager journal log. ▪ all messages (**0**) ▪ warning, error, and fatal messages (**1**) ▪ error and fatal messages (**2**) ▪ fatal messages only (**3**) The default level is (**1**).
Logging level	Specifies the threshold severity level at which the system issues informational, warning, and error messages, and writes them to the JMS IQ Manager log. ▪ all messages (**0**) ▪ warning, error, and fatal messages (**1**) ▪ error and fatal messages (**2**) ▪ fatal messages only (**3**) The default level is (**1**).
Max. log file size	Specifies the maximum size of the JMS IQ Manager log file, in MB. The default is **10** MB.
Message performance interval	Specifies the number of messages the JMS IQ Manager processes before logging its performance statistics. The default is **1000**.
Number of backup log files	Specifies the maximum number of JMS IQ Manager backup log files in the stack. The default is **5**.
Performance interval	Specifies the number of IMessages the IMessageManager creates before logging its performance statistics. The default is **1000**.

Stable Storage

The Stable Storage properties allow you to configure the storage and journaling operations for the JMS IQ Manager. See the *eGate Integrator JMS Reference Guide* for additional information.

Figure 320 Stable Storage Configuration Properties

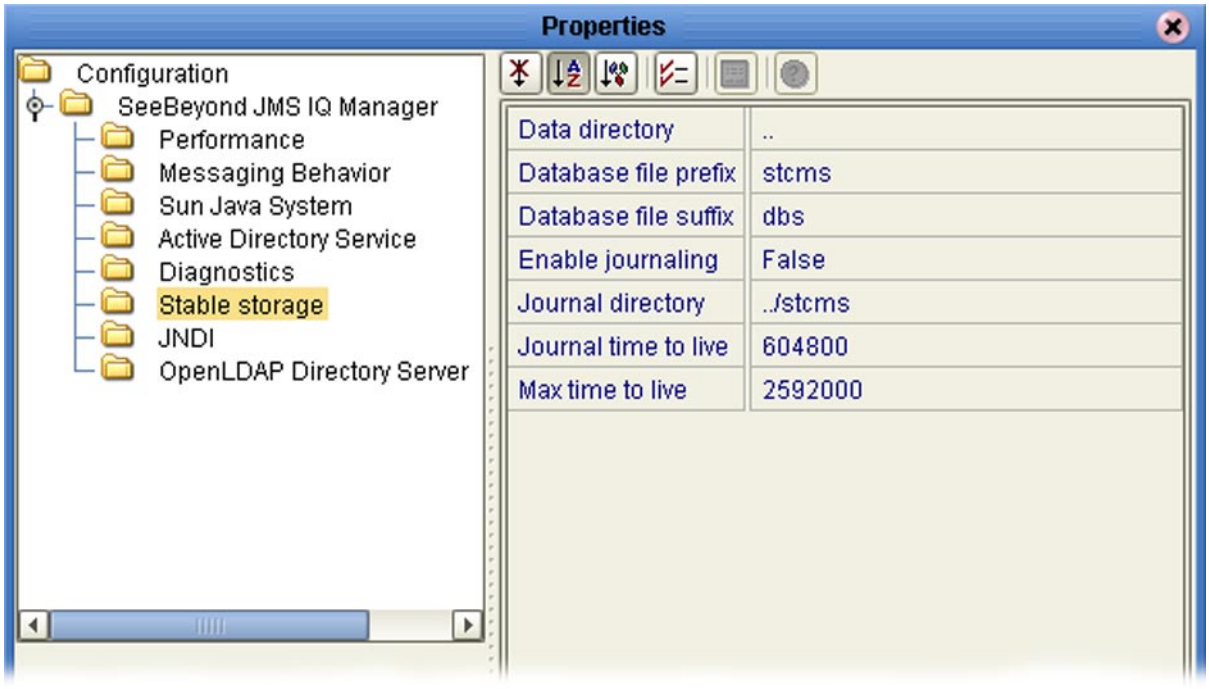


Table 116 Stable Storage Configuration Properties List

Property	Description
Data directory	Specifies the location for the JMS IQ Manager database files. The path can be either absolute or relative to the **logicalhost** directory. The default is **..** (one directory up from the **logicalhost** directory).
Database file prefix	Specifies the prefix for the JMS IQ Manager database file names. The default is **stcms** (for default file names of *stcms*.dbs*).
Database file suffix	Specifies the extension for the JMS IQ Manager database file names. The default is **dbs** (for default file names of *stcms*.dbs*).
Enable journaling	Enables or disables journaling, whereby every inbound message is automatically copied to the journal database. **True** enables, **False** disables; the default is **False**.

Table 116 Stable Storage Configuration Properties List

Property	Description
Journal directory	Specifies the location for the journal database files. The path can be either absolute or relative to the **logicalhost** directory. The default is **../stcms**.
Journal time to live	Specifies the maximum amount of time, in seconds, a journaled message persists before expiring. The default is **604,800** seconds (7 days).
Max. time to live	Specifies the maximum amount of time, in seconds, a live message persists before being removed from the queue. The default is **2,592,000** seconds (30 days).

JNDI

The Java Naming and Directory Interface (JNDI) configuration properties are not user-definable.

OpenLDAP Directory Server

These configuration properties are used only if you are using the OpenLDAP Directory Server as an LDAP server. See the *eGate Integrator System Administration Guide* for additional information.

Figure 321 OpenLDAP Directory Server Configuration Properties

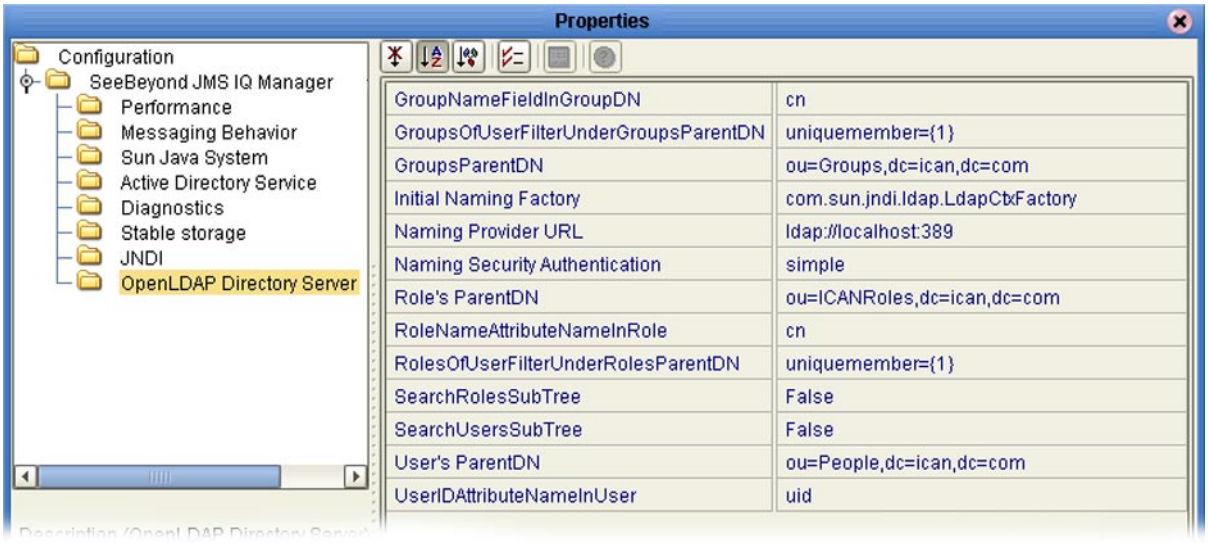


Table 117 OpenLDAP Directory Server Configuration Default Properties List

Property	Default Value
GroupNameFieldInGroupDN	cn
GroupOfUserFilterUnderGroupsParentDN	uniquemember={1}
GroupsParentDN	ou=Groups,dc=ican,dc=stc.com
Initial Naming Factory	com.sun.jndi.ldap.LdapCtxFactory
Naming Provider URL	ldap://localhost:389
Naming Security Authentication	simple
Role's ParentDN	ou=ICANRoles,dc=ican,dc=com
RoleNameAttributeNameInRole	cn
RolesOfUserFilterUnderGroupsParentDN	uniquemember={1}
SearchRolesSubTree	False
SearchUsersSubTree	False
User's ParentDN	ou=People,dc=ican,dc=com
UserIDAttributeNameInUser	uid

Chapter 10

Project Deployment

This chapter describes the process of creating deployment profiles and activating the deployed projects.

What's in This Chapter

10.1 Deployment Profiles

Deployment Profiles define specific instances of a Project in a particular Environment. A deployment profile contains information about the assignment of Services and Message Destinations to integration and message servers (JMS IQ Managers). It also contains version information for all relevant objects in the Project. The Enterprise Designer includes a Deployment Editor, which you can use to create and customize deployment profiles.

Note that:

- Each Project can have zero or more Deployment Profiles, but each of a Project's active Deployment Profiles must be in a separate Environment.
- Each Environment can have zero or more Deployment Profiles assigned to it, but any given Environment can have only one Deployment Profile from a given Project.

Project deployment is illustrated in **Figure 2 on page 36**.

10.2 Deployment Editor

The Deployment Editor (see Figure 322) allows you to create a new Deployment Profile or edit an existing one. To create a new Deployment Profile, right-click on a Project in the Project Explorer to display its context menu. From the menu, select **New > Deployment Profile**. To edit an existing Deployment Profile, simply click on its icon.

Figure 322 Deployment Editor Window

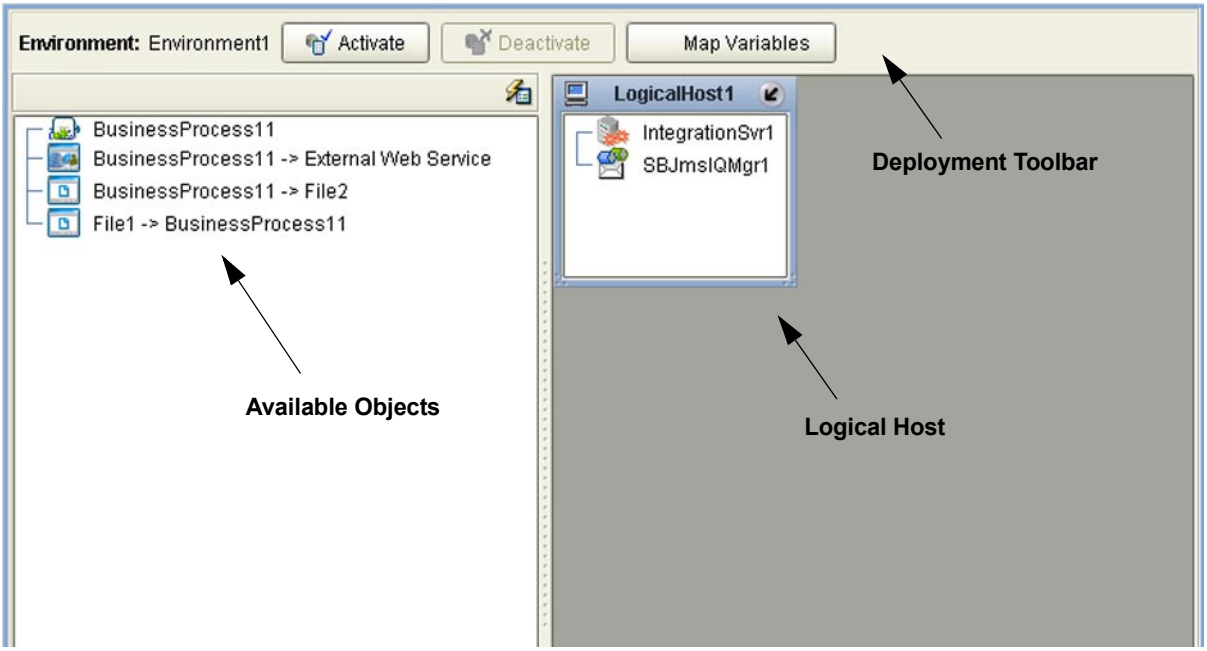


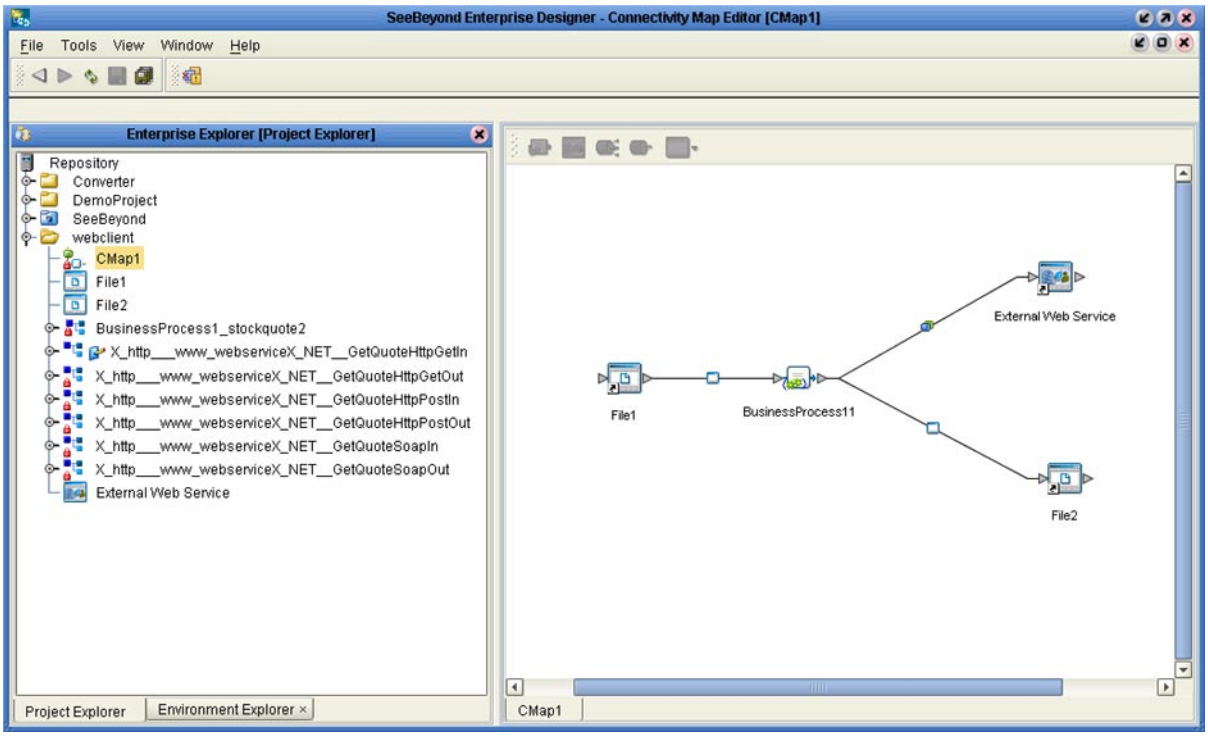
Table 118 Deployment Editor Buttons and Icons

Button/Icon	Function
Activate Reactivate	Starts the Project by creating an enterprise archive (EAR) file based on the Connectivity Map and linking this file with the application server. After activation, button changes to **Reactivate**. See **Activating and Deactivating Deployment Profiles** on page 403.
Deactivate	Stops the Project by terminating the link between the EAR file and the application server, sets the Deployment Profile to *inactive*, and saves to the Repository. See **Activating and Deactivating Deployment Profiles** on page 403.
Map Variables	Allows you to assign names and values to Project variables for the specific Deployment Profile. See **Mapping Variables** on page 401.
Auto Map	Automatically deploys components to their matching containers, when there is a one-to-one correspondence between them. See **Automapping** on page 400.

10.3 Creating a Deployment Profile

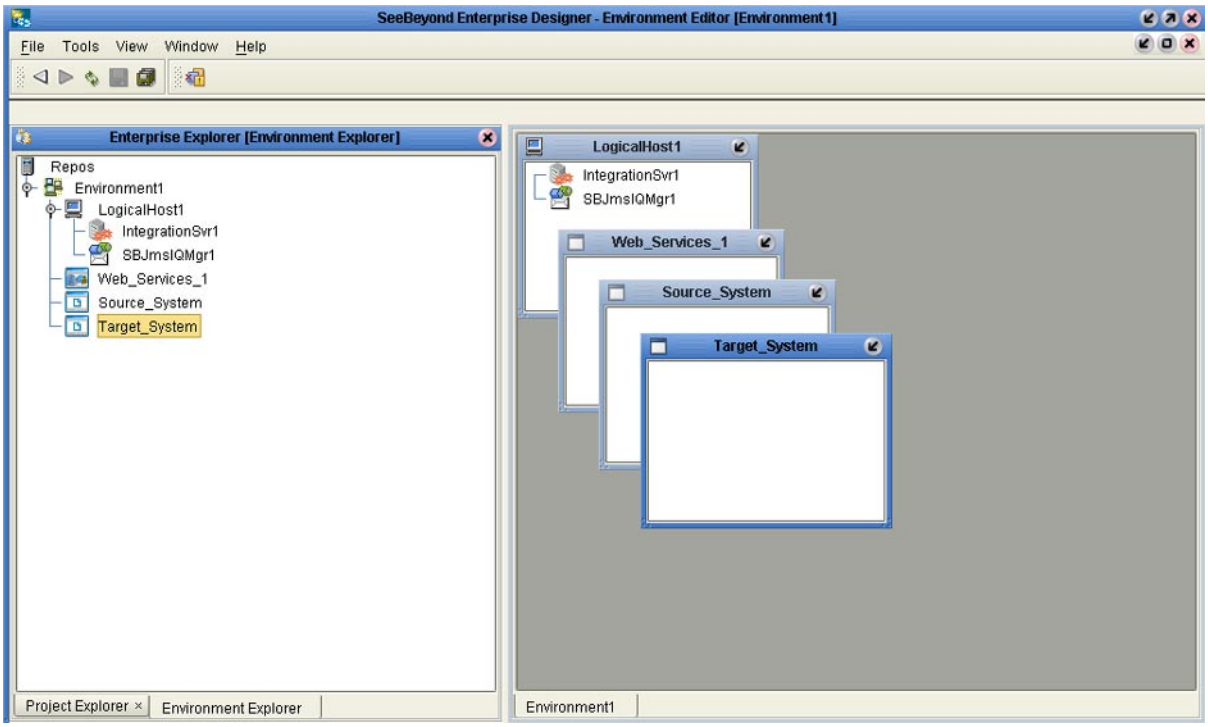
The Web Client Project shown in Figure 323 will be used as a deployment example.

Figure 323 Web Client Example Project



1. In the Environment Explorer, right-click on the Repository to display its context menu.
2. Select **New Environment**, and assign an appropriate name.
3. Right-click on the Environment to display its context menu, and create the components you need, for example:
   - A A new Logical Host
   - B A new Integration Server
   - C A new JMS IQ Manager
   - D A new External Web Service
4. Name the Environmantal components appropriately. They will appear as shown in Figure 324.

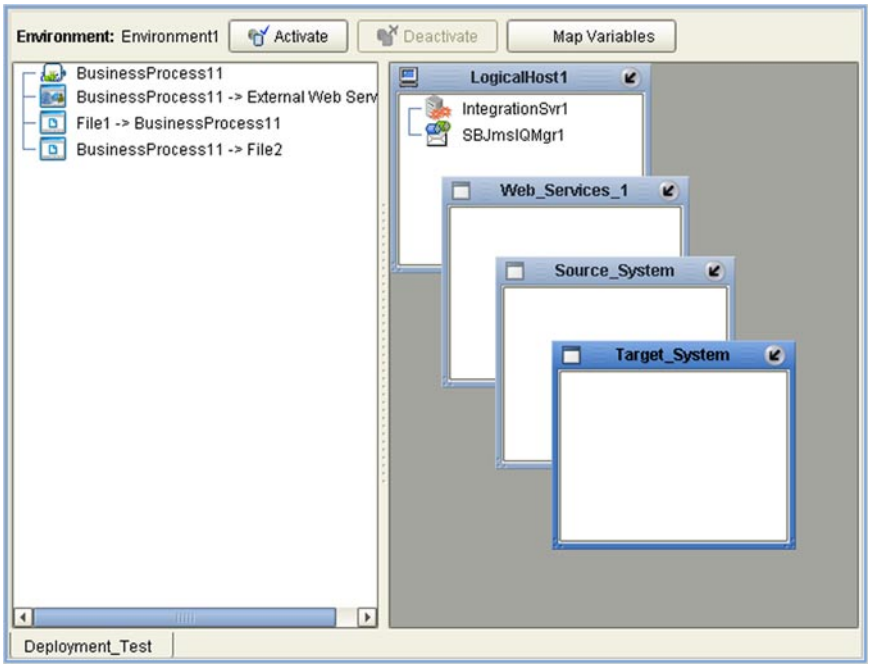
Figure 324 Web Client Example Environment



5 In the Project Explorer, right-click on the Project to display its context menu.

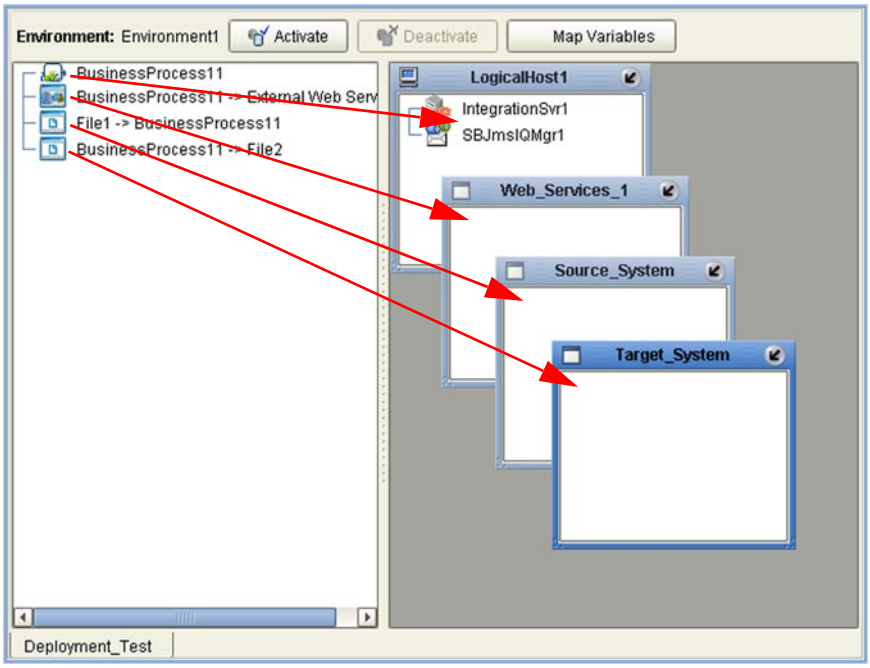
6 From the menu, select **New > Deployment Profile**. The Deployment Profile Editor appears, displaying the Environment you created previously (see Figure 325).

Figure 325 Example Deployment Profile (1)



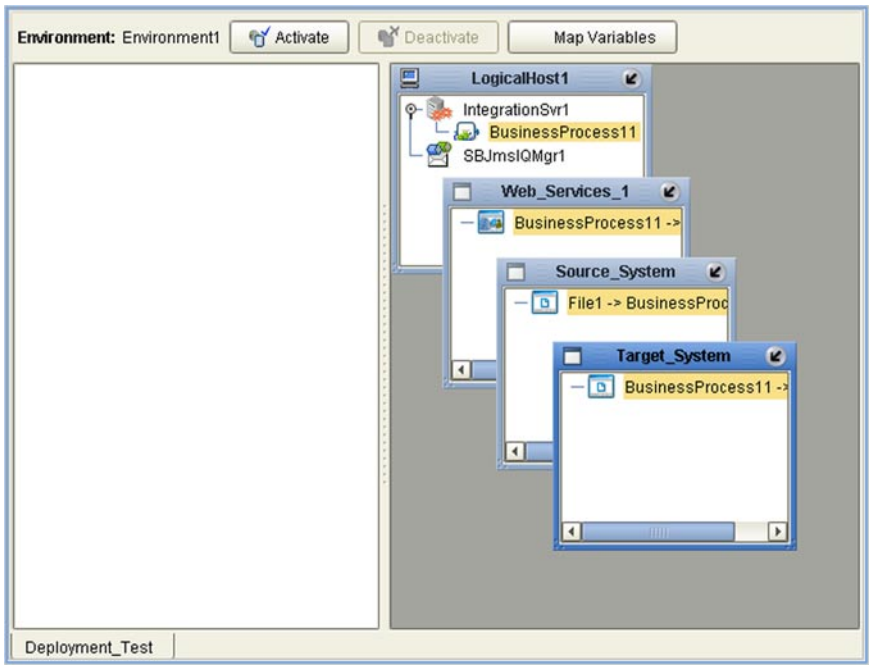
7 Drag the Project components from the left panel and drop them into the appropriate Environment components in the right panel, as illustrated in Figure 326.

Figure 326 Example Deployment Profile (2)



8 When the Environment components are fully populated, the left panel will be blank, as shown in Figure 327. You should now **Save** the profile.

Figure 327 Example Deployment Profile (3)



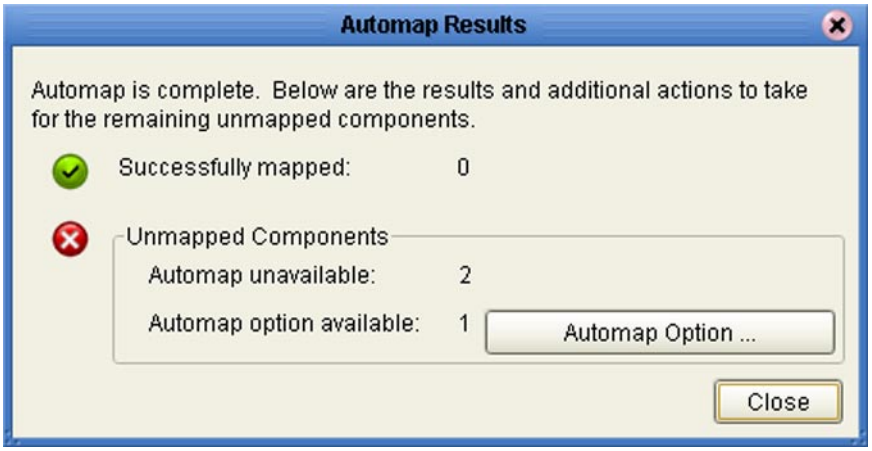
10.4 Automapping

When a one-to-one correspondence exists between the available objects and the containers in the Environment to which you want to deploy them, clicking the **Auto Map** icon (see Figure 328) automatically deploys the components to their matching containers. (This feature only works with external systems for which it is enabled.)

Figure 328 Auto Map Icon

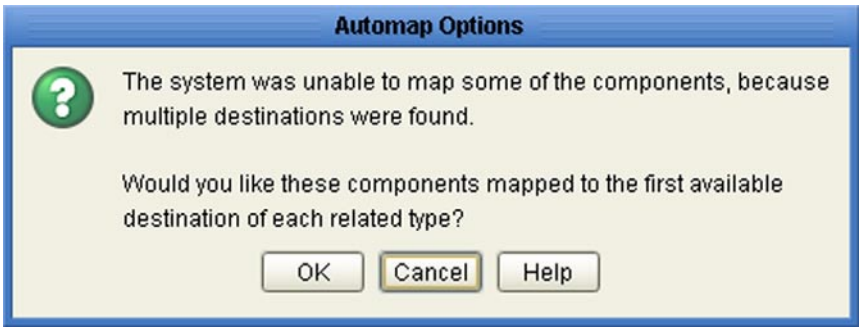
After the Automap feature executes, a dialog box is displayed showing the results (see Figure 329). The lack of an appropriate container, or ambiguity between potential containers, will cause a component to remain unmapped.

Figure 329 Automap Results Dialog Box



In the case of an ambiguity, for example, mapping options are presented in an Automap Options dialog box (see Figure 330) that is displayed when you click **Automap Option**. You may click **OK** to accept the Automapping option, or click **Cancel** and map the deployable components manually.

Figure 330 Automap Options Dialog Box



10.5 Mapping Variables

Project variables function as placeholders, having values that are determined when you create a specific Deployment Profile. These values can be literals or Environmental constants. Clicking the **Map Variables** button displays the Deployment Profile Mappings panel, where you can assign names (see Figure 332) and values (see Figure 333).

Figure 331 Map Variables Button

Figure 332 Deployment Profile Mappings

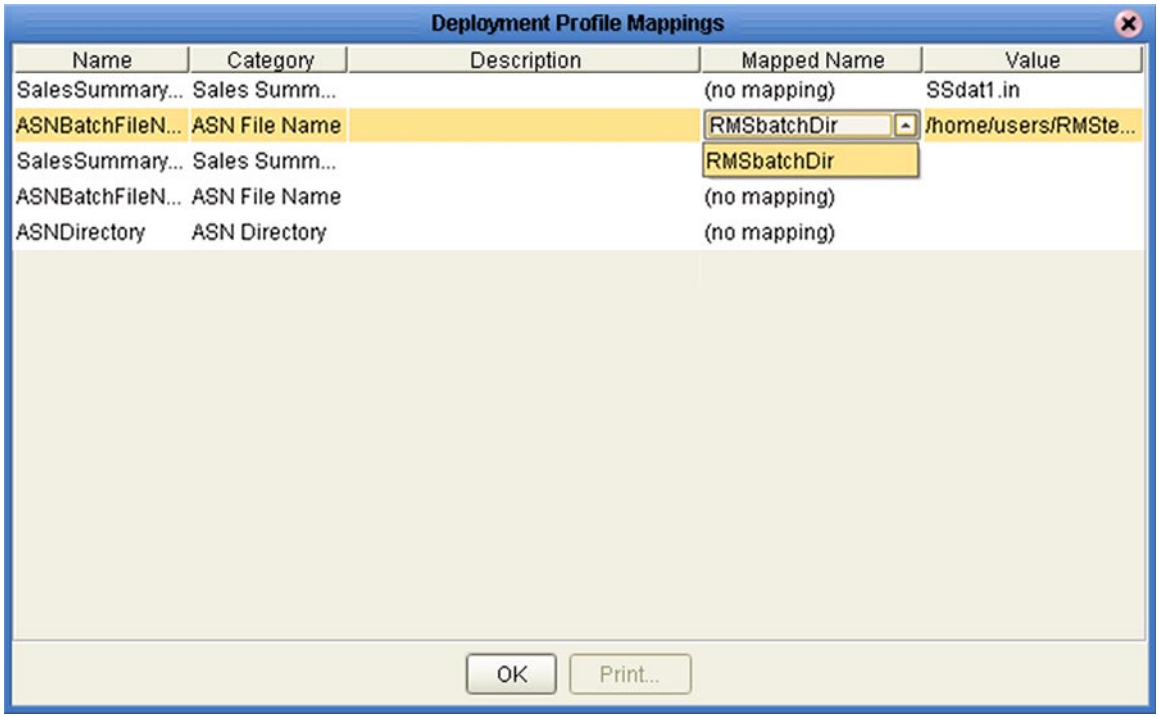
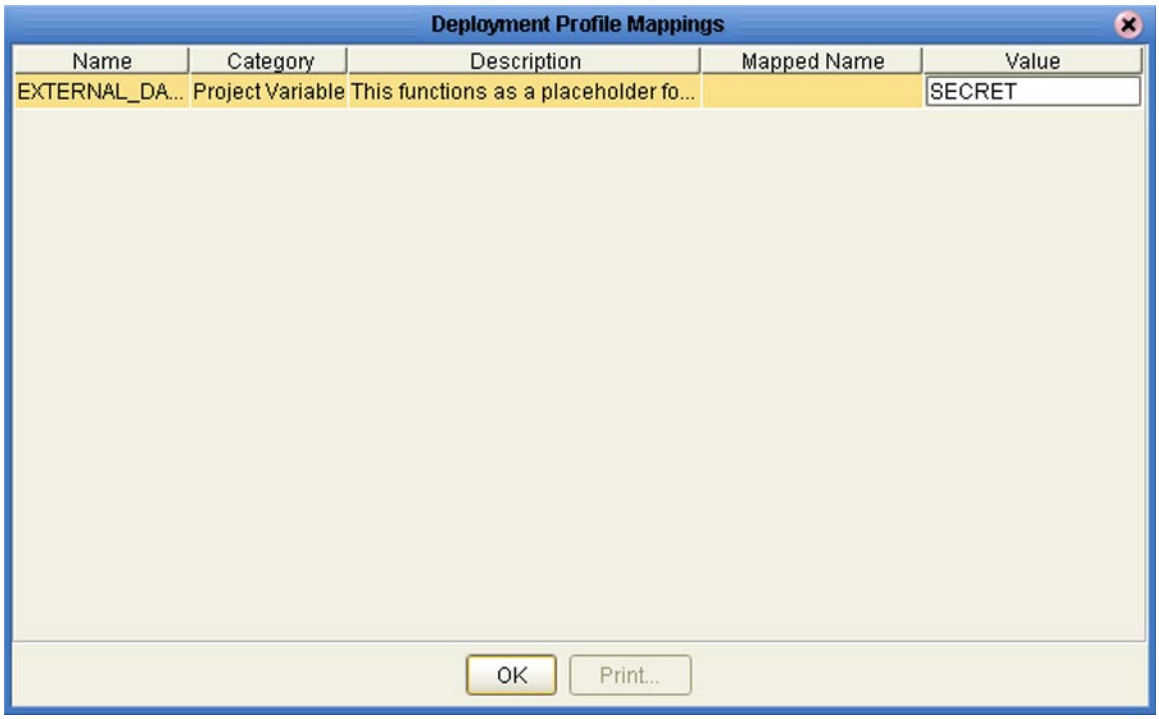


Figure 333 Project Variable Value Entry



10.6 Activating and Deactivating Deployment Profiles

When activating or deactivating a Deployment Profile, you have the option of applying the changes to the Logical Host either immediately or at a later time. By activating the Deployment Profile without immediately applying the changes, you can check the validity of the entire Deployment Profile.

If you have multiple Deployment Profiles to deploy, you might work more efficiently by activating each of the Deployment Profiles without applying the changes, and then applying all of the changes to the Logical Host at a later time (*heed the warning given in the following procedures*).

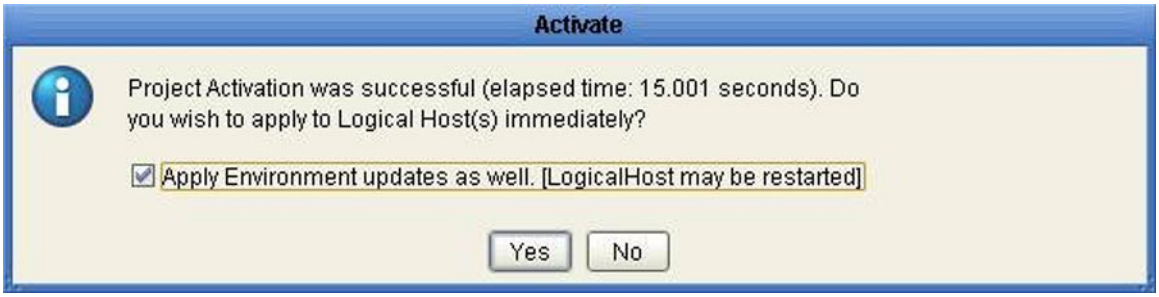
10.6.1 Using Enterprise Designer

To activate a Deployment Profile

1 In the Project Explorer, select the Deployment Profile you want to activate.

2 Click the **Activate** button, which invokes the dialog box shown in Figure 334:

Figure 334 Activate Dialog Box



3 Select your response based on the following criteria:

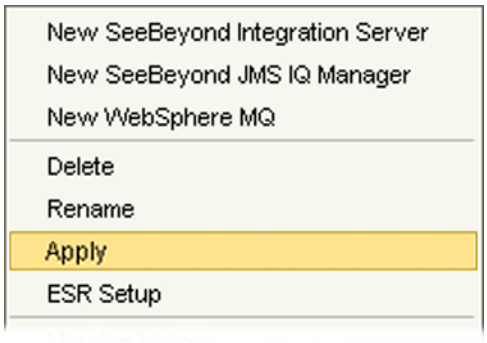
- If the Logical Host is running, and you wish to apply the changes immediately, click **Yes**.
- If the Logical Host has not yet been bootstrapped, or you wish to apply the changes at a later time, click **No**.

4 If you click **Yes**, the information box shown in Figure 335 will appear after the changes have been applied. Click **OK** to proceed.

Figure 335 Success Information Box

5 To apply the changes at a later time, right-click the Logical Host and select **Apply** from the menu (see Figure 336). This will download the latest configuration from the Repository to that Logical Host.

Figure 336 Logical Host Context Menu - Apply



6 The information box shown in Figure 335 will appear after the changes have been completed. Click **OK** to proceed.

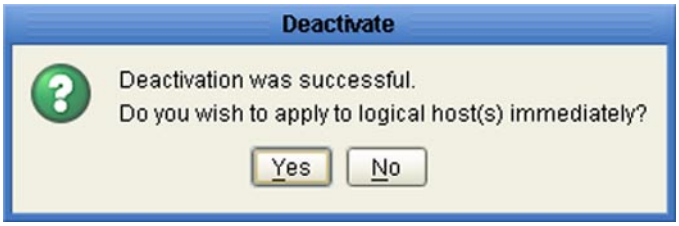
Important: *Use the ICAN Monitor to verify that your changes have been applied before again running the apply command.*

To deactivate a Deployment Profile

1 In the Project Explorer, select the Deployment Profile you want to deactivate.

2 Click the **Deactivate** button, which invokes the dialog box shown in Figure 337:

Figure 337 Deactivate Dialog Box



3 Select your response based on the following criteria:

- If the Logical Host is running, and you wish to apply the changes immediately, click **Yes**.
- If the Logical Host has not yet been bootstrapped, or you wish to apply the changes at a later time, click **No**.

4 To apply the changes at a later time, right-click the Logical Host and select **Apply** from the menu (see Figure 336). This will apply all of the changes for that Logical Host.

Important: *Use the ICAN Monitor to verify that your changes have been applied before again running the apply command.*

10.6.2 Using a Command-line Script

A script named **CmdLineUtil.bat** (or **.sh**) allows you to deploy and undeploy projects via the command line. The *apply* and *unapply* commands provide the same function as clicking **Activate** or **Deactivate**, and then clicking **Yes**, in Enterprise Designer.

This command-line utility allows you three options for deployment:

- **Apply to all Logical Hosts in the Environment** (you must provide the Environment name).
- **Apply to a single Logical Host** (you must provide the Environment name and the Logical Host name).
- **Apply to a specific Deployment Profile** (you must provide the Project name and the Deployment Profile name).

The command-line utility also allows you to activate or deactivate from any computer, not only the one on which the Logical Host is installed. User name/password authentication is performed before the utility can be used.

This utility is downloaded using Enterprise Manager as described in the *SeeBeyond ICAN Suite Installation Guide*. Before running the utility, you must perform the following tasks:

- Set up the JAVA_HOME environmental variable on the host computer.
- Start the Repository.
- Activate or deactivate the Project in Enterprise Designer.
- Perform an initial bootstrap of the Logical Host(s) to download the components from the Repository (see **Bootstrapping** on page 356).

Important: *Use the ICAN Monitor to verify that your changes have been applied before again running the apply or unapply command.*

10.7 Deploying Projects to Third-Party Servers

SeeBeyond's eGate Integrator allows you to develop Projects using Enterprise Designer and deploy them to a BEA WebLogic or IBM WebSphere environment. The SAR files for these third-party products must be installed prior to deployment, as described in the *eGate Integrator JMS Reference Guide*.

Because of the versions of the Java Connection Architecture supported by WebLogic and WebSphere, the following restrictions apply:

- Services deployed to WebLogic or WebSphere are restricted to those internal to eGate Integrator itself (between message destinations), and those associated with outbound eWays.
- Not all SeeBeyond eWays support third-party servers. Check the individual eWay User's Guides regarding such support, and also any additional configuration that may be necessary for compatibility with WebLogic or WebSphere.

10.7.1 BEA WebLogic

Note: *Before using the WebLogic JMS, you must install additional* ***.jar*** *files as described below. For additional information, see the eGate Integrator JMS Reference Guide.*

To install additional .jar files

1. Download the **log4j.jar** file from the location below (location subject to change).

   http://jakarta.apache.org/log4j/docs
2. Download the **xerces.jar** file from the location below (location subject to change).

   http://xml.apache.org/dist/xerces-j
3. Copy the following files from ***ICAN-root*****\edesigner\usrdir\modules**:
   - **com.stc.eventmanagementapi.jar**
   - **com.stc.eventmanagementimpl.jar**
   - **com.stc.jms.stcjms.jar**
4. Place all **.jar** files into the ***ICAN-root******weblogic8x*****\server\lib** directory.
5. Add the **.jar** files to the *set CLASSPATH* segment of the **startWLS.cmd** file located in the ***ICAN-root******weblogic8x*****\server\bin** directory. The text to be added is:

```
%WL_HOME%\server\lib\log4j.jar;%WL_HOME%\server\lib\xerces.jar;
%WL_HOME%\server\lib\ com.stc.eventmanagementapi.jar;
%WL_HOME%\server\lib\ com.stc.eventmanagementimpl.jar;
%WL_HOME%\server\lib\ com.stc.jms.stcjms.jar
```

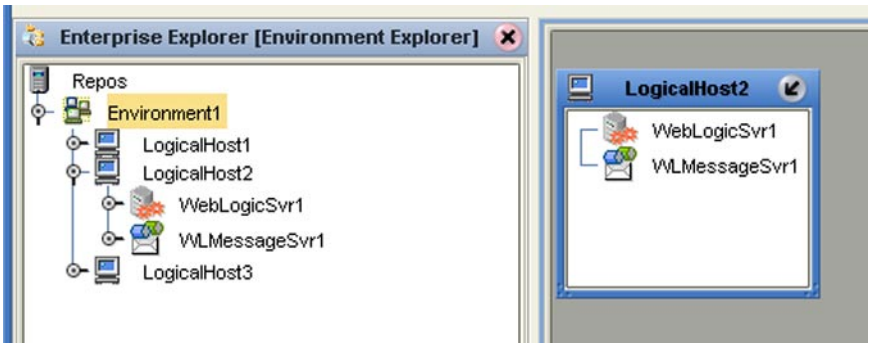
To deploy an ICAN Project to a BEA WebLogic 8.0 or 8.1 environment

1. Create the following components in Enterprise Designer (see Figure 338):
   - A A new environment
   - B A Logical Host

C A WebLogic J2EE application server

D A WebLogic JMS message server

Figure 338 WebLogic Deployment (1)

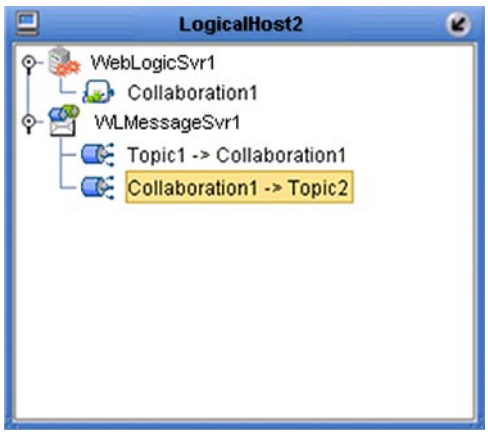


2 Create a new Deployment Profile to bind the Connectivity Map to the new WebLogic environment (see Figure 339).

A Drag the two topics and drop onto the WebLogic message server.

B Drag the Collaboration and drop onto the WebLogic application server.

Figure 339 WebLogic Deployment (2)



3 Activate the Deployment Profile.

Activating the Deployment Profile creates an Environment Archive (EAR) file, which contains all files necessary to create and run an application in WebLogic. This file can be found in the following location:

ICAN-root`\repository\data\files\`***`WLEnvironmrntName`***`\`
`ProjectName_DeploymentProfileName``.ear`

Note: *The remainder of this procedure is performed in the WebLogic user interface, and is only outlined here. Please refer to your BEA WebLogic documentation for current information regarding interface layout and deployment details.*

4 Start the BEA WebLogic server.

5 Navigate to **Server Administration Console > Deployments > Applications**.

6 Perform the following steps:

A Add a new JMS Connection Factory.

B Enter a JNDI name for the JMS Connection Factory:

```
jms\connectionfactory\xa-topic\
     LogicalHostName_MessageServerName
```

For example, the default name would be:

```
jms\connectionfactory\xa-topic\LogicalHost1_WLMessageSvr1
```

C Verify that the WebLogic JMS Server Destination names for topics match those in eGate Integrator.

D Select **Deploy a new Application**.

E Upload and install the EAR file described in step 3.

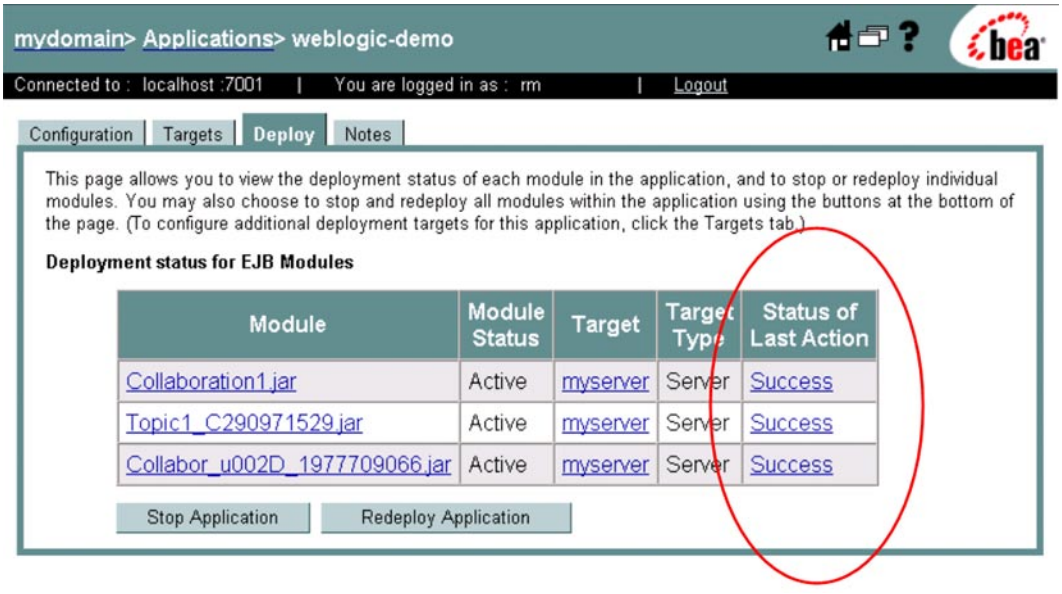
F Select the EAR file you just installed as the archive for the new application.

G Enter a name for the new application.

H Click **Deploy**.

I Verify the success of the deployment (see Figure 340, which shows a WebLogic 8.1 example).

Figure 340 WebLogic Deployment Verification



10.7.2 IBM WebSphere

Note: *Before using the WebSphere JMS, you must install additional* ***.jar*** *files as described below. For additional information, see the eGate Integrator JMS Reference Guide.*

To install log4.jar

1 Download the **log4j.jar** file from the location below (location subject to change).

http://jakarta.apache.org/log4j/docs

2 Copy the following files from ***ICAN-root*\edesigner\usrdir\modules**:

- **com.stc.antlrimpl.jar**
- **com.stc.eventmanagementapi.jar**
- **com.stc.eventmanagementimpl.jar**
- **com.stc.jms.stcjms.jar**

3 Place all **.jar** files into the ***ICAN-root*\WebSphere\AppServer\lib** directory.

To deploy an ICAN Project to an IBM WebSphere 5.0.x or 5.1 environment

1 Create the following components in Enterprise Designer (see Figure 341):

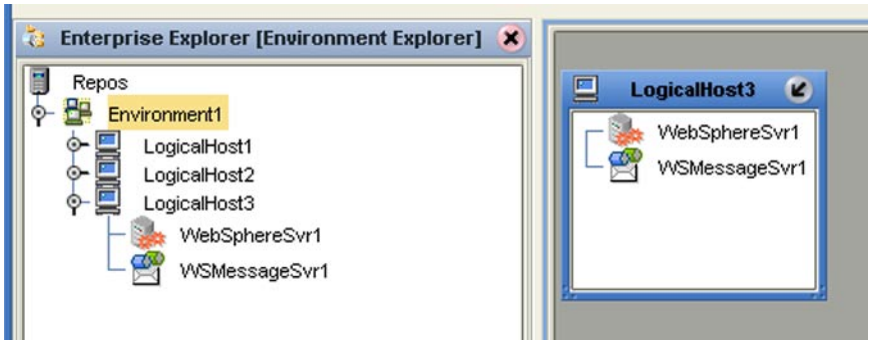
A A new environment

B A Logical Host

C A WebSphere J2EE application server

D A WebSphere JMS message server

Figure 341 WebSphere Deployment (1)

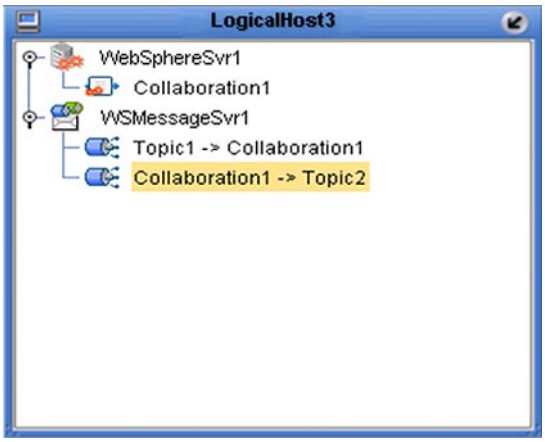


2 Create a new Deployment Profile to bind the Connectivity Map to the new WebSphere environment (see Figure 342).

A Drag the two topics and drop onto the WebSphere message server.

B Drag the Collaboration and drop onto the WebSphere application server.

Figure 342 WebSphere Deployment (2)



3 Activate the Deployment Profile.

The activated Deployment Profile creates an Environment Archive (EAR) file, which contains all files necessary to create and run an application in WebSphere. This file can be found in the following location:

ICAN-root\repository\data\files***WSEnvironmentName***\
ProjectName_DeploymentProfileName.ear

Note: *The remainder of this procedure is performed in the WebSphere user interface, and is only outlined here. Please refer to your IBM WebSphere documentation for current information regarding interface layout and deployment details.*

4 Start the IBM WebSphere server.

5 From the Administrative Console, navigate to **Servers > Application Servers > *server_name* > Message Listener Service > Listener Ports**.

6 Add a new Listener port.

7 Enter a Connection Factory JNDI name for the new port:

jms\connectionfactory\xa-topic\
LogicalHostName_MessageServerName

For example, the default name would be:

jms\connectionfactory\xa-topic***LogicalHost1_WSMessageSvr1***

This binds the JNDI name with the WebSphere Message Server Listener port.

8 From the Administrative Console, navigate to **Applications > Enterprise Applications > Install New Application**.

9 In *Preparing for the application installation*:

A Enter the path for the EAR file described in step 3 and click **Next**.

B Select **Generate Default Bindings** and click **Next**.

10 In *Step 1, Provide options…*:

A Check **Deploy EJBs**.

B Enter the application name.

C Click **Next**.

11 In *Step 2, Provide options…*, click **Next**.

12 In *Step 3, Provide Listener Ports…*, accept the default value and click **Next**.

Note: *The Listener port number should match the port number entered in step 6.*

13 In *Step 4, Provide JNDI Names…*, accept the default value and click **Next**.

14 In *Step 5, Provide EJB references…*, accept the default value and click **Next**.

15 In *Step 6, Map resource references…*, enter the JNDI name from step 7, and click **Next**.

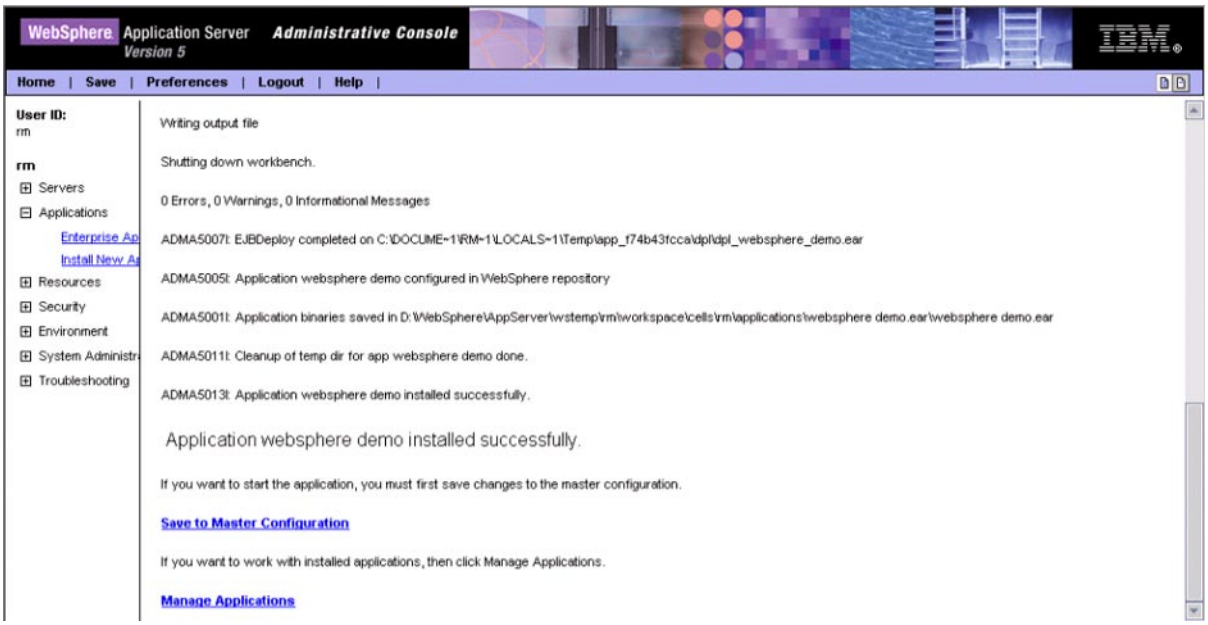
16 In *Step 7, Map modules…*, check all modules and click **Next**.

17 In *Step 8,* (protection levels), check all modules and click **Next**.

18 In *Step 9, Summary,* click **Finish**.

19 Verify the success of the deployment (see Figure 343, which shows a WebSphere 5 example).

Figure 343 WebSphere Deployment Verification



Chapter 11

Web Services

This chapter describes the use of the Web Services capability of eGate Integrator, acting with other components of the ICAN Suite.

What's in This Chapter

11.1 Overview

Basically, Web Services enables communication and data transfer between diverse applications using the Internet. In doing so, it provides a means for implementing EAI (Enterprise Application Integration) within an organization, or B2B (Business-to-Business) integration between partner organizations. This capability is achieved by wrapping back-end systems to present a common, standardized interface to the connecting network.

Four related technologies are used to transform and transport data within Web Services:

- **XML** (Extensible Markup Language)

  Provides a language for defining both the data itself and the way to process it.

- **WSDL** (Web Services Description Language)

  Defines the interfaces, data types, interactions, and mappings used in the Web Services. WSDL files, which are XML-based, are used to invoke and operate Web Services on the Internet, and to access and invoke remote applications and databases.

- **SOAP** (Simple Object Access Protocol)

  Defines a communications envelope that is mappable to HTTP and provides a format for transmitting XML documents over a network.

- **UDDI** (Universal Description, Discovery, and Integration)

  Provides a mechanism for storing and categorizing information that allows publication of services and discovery of external services.

11.2 ICAN Web Services

eGate Integrator provides the capability to create either a Web Service, which can be invoked by a remote client, or a Web Services Client to invoke a Web Service. A Business Process developed in eInsight Business Process Manager, for example, can be exposed as a Web Service. As such, it can invoke other external Web Services or be invoked by any Web Services Client. See **Building a Web Services Client** on page 428 and **Creating a Web Service** on page 423.

The ICAN Suite contains the following components that implement the Web Services capability:

- **WSDL Wizard**

  The WSDL Wizard creates an OTD from a WSDL file. See **Using the WSDL Wizard** on page 141.

- **WSDL Editor**

  See the *eInsight Business Process Manager User's Guide*.

- **WSDL Interface Designer**

  See the *eInsight Business Process Manager User's Guide*.

- **WSDL Viewer**

  See the *eInsight Business Process Manager User's Guide*.

- **UDDI Registry**

  All objects represented in the ICAN Repository that can be accessed as Web Services are presented via a UDDI registry. See **UDDI Registry** on page 414.

ICAN Web Services adheres to guidelines and specifications developed by the Web Services-Interoperability Organization (WS-I) to the maximum possible extent. As a result, ICAN Web Services currently supports *document/literal* and *RPC/literal* messaging only, since the WS-I Basic Profile 1.1 rules out the use of *encoded* messaging.

11.3 UDDI Registry

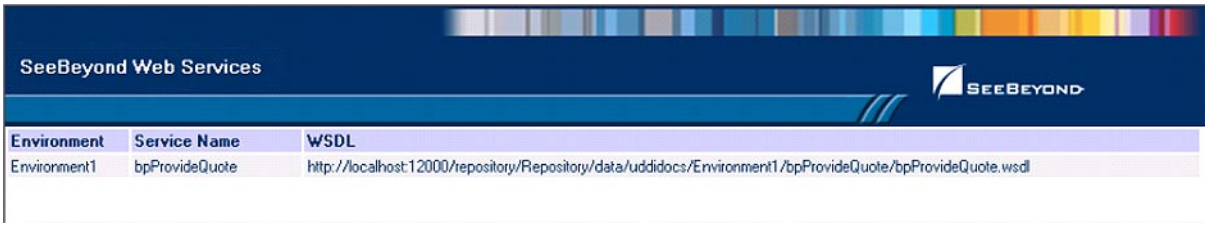
In general, all ICAN objects that expose themselves as Web Services, such as eInsight Business Processes, are listed in a UDDI registry. The deployment activation process automatically publishes entries to the UDDI registry and creates the necessary sections in the WSDL files to expose them as Web Services.

This registry can be viewed on the SeeBeyond Web Services page (see Figure 344), part of Enterprise Manager. The URL of the SeeBeyond Web Services page is:

```
http://hostname:portnumber/stcuddi
```

*Note: The **hostname** is the fully-qualified, network-addressable host name of the server where you installed the Repository. The **portnumber** is the number of the port you entered during installation of the Repository.*

Figure 344 SeeBeyond Web Services Page



Each entry on the SeeBeyond Web Services page includes:

- The ICAN environment name.
- The actual (Web) Service name.
- The URL for the associated WSDL file.

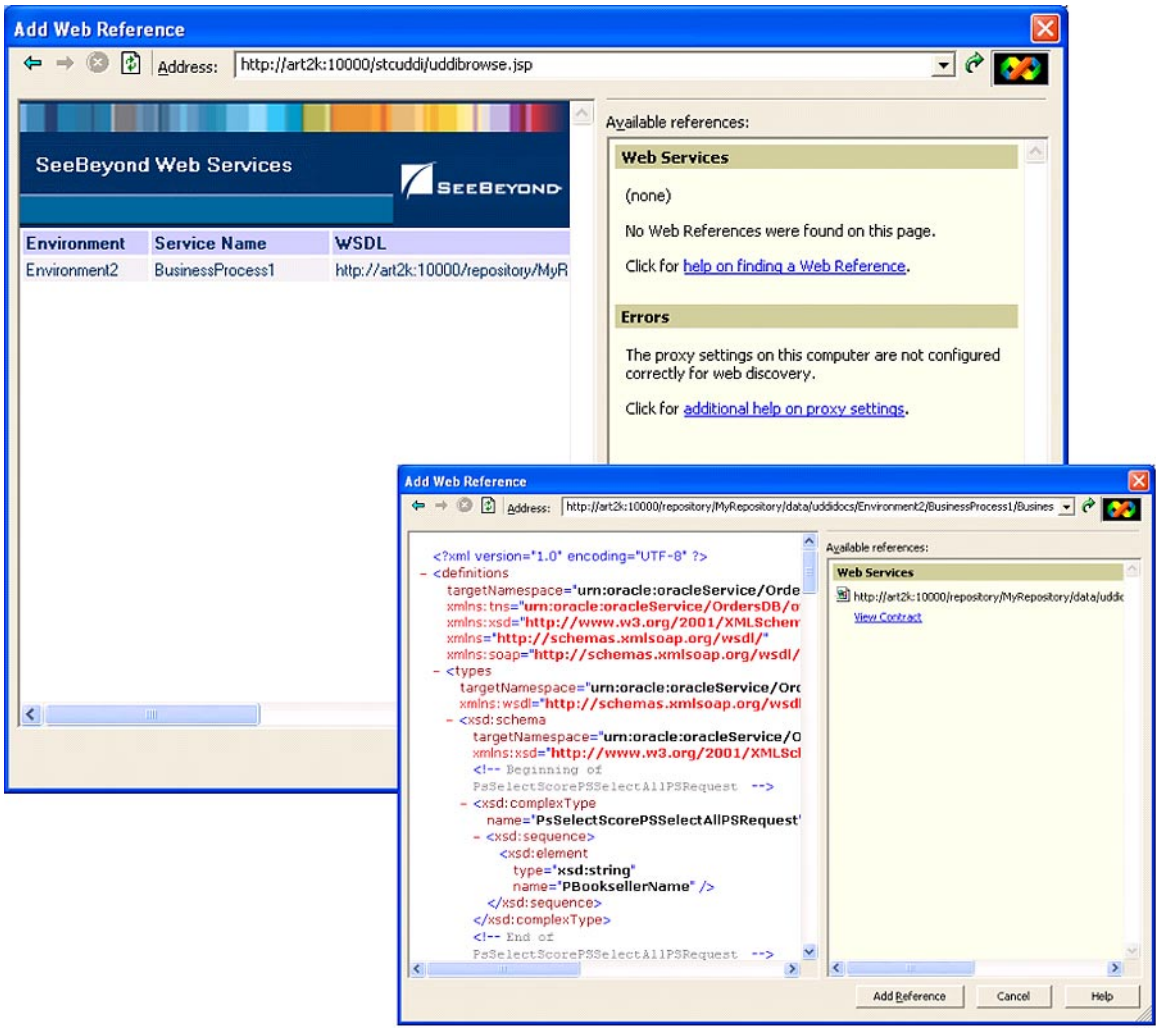
Select an entry (line item) to display its WSDL file, as shown in Figure 345.

Figure 345 Example Web Service WSDL File

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions targetNamespace="http://seebeyond/quoteservice" xmlns:tns="http://seebeyond/quoteservice"
    xmlns:sbynpx="http://bpel.seebeyond.com/hawaii/5.0/privateExtension/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  - <types targetNamespace="http://seebeyond/quoteservice">
    - <xsd:schema targetNamespace="http://seebeyond/quoteservice"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:element type="xsd:string" name="QuoteRequest" />
        <xsd:element type="xsd:string" name="QuotePrice" />
        <xsd:element type="xsd:string" name="QuoteCompany" />
      </xsd:schema>
    </types>
  - <message name="msgProvideQuoteRequest">
      <part name="TICKER" element="tns:QuoteRequest" />
    </message>
  - <message name="msgQuoteResponse">
      <part name="price" element="tns:QuotePrice" />
      <part name="company" element="tns:QuoteCompany" />
    </message>
  - <portType name="ptProvideQuote">
```

The SeeBeyond UDDI Registry can be used in a third party tool, for example Microsoft Visual Studio (see Figure 346). In Visual Studio's Solution Explorer, right-click on *References* and enter the URL of the SeeBeyond Web Services page as a Web Reference.

Figure 346 Microsoft Visual Studio Example



eGate Integrator can exchange data over the Internet with Web Services applications described using the Web Services Description Language (WSDL). This language is used to define Web Services and describe how to access them. The WSDL OTD Wizard is used to build OTDs that are used in the Project Collaborations (see **Using the WSDL Wizard** on page 141).

11.3.1 Using UDDI Browsers

A third-party UDDI Browser can be used to access the SeeBeyond UDDI Registry for publishing and inquiry. To add the SeeBeyond UDDI Registry to the browser, you need to enter the information given in Table 119 into the browser's *Add UDDI Registry* facility.

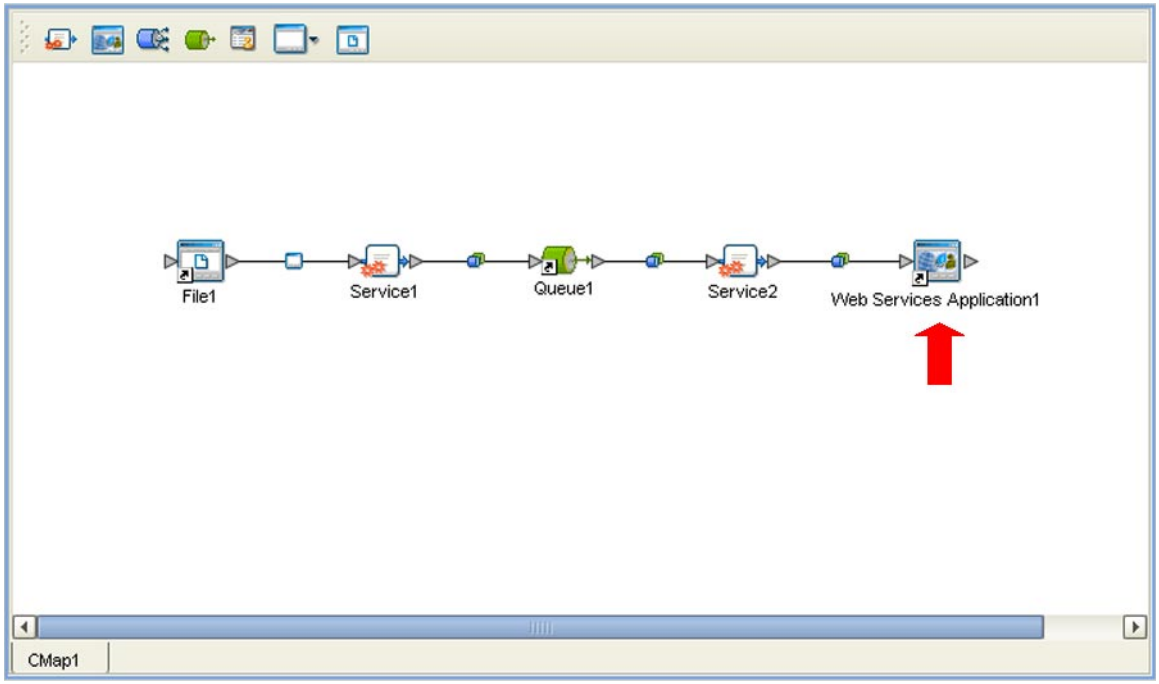
Table 119 UDDI Registry Information

Parameter	Value
Name	SeeBeyond UDDI
Inquiry URL	http://***hostname:portnumber***/stcuddi/inquiry
Publish URL	http://***hostname:portnumber***/stcuddi/publish
Username	***your_username***
Password	***your_password***

11.4 Web Services Application

A Web Services Application (see Figure 347) represents either an external Web Service that can be invoked by a Web Services Client, based on an eInsight Business Process, or a Business Process exposed as a Web Service. In the case of the latter, the Web Service is listed in the SeeBeyond UDDI Registry.

Figure 347 Enterprise Designer - Connectivity Map Editor



If you are creating more than one Web Services Application for deployment in any given Environment, you must ensure that the respective *servlet context* properties have different values so that each application will have a different URL. See **Web Services External System** on page 418.

11.5 Web Services External System

A Web Services External System represents a system containing a Web Services Application. The Properties dialog box (see Figure 348) appears automatically when you create a New Web Services External System in the Environment Explorer. You can also invoke the dialog box by right-clicking the Web Services Application in the Enterprise Explorer and selecting **Properties**. You should assign the desired values for the listed properties.

Figure 348 Web Service External System Properties Dialog Box

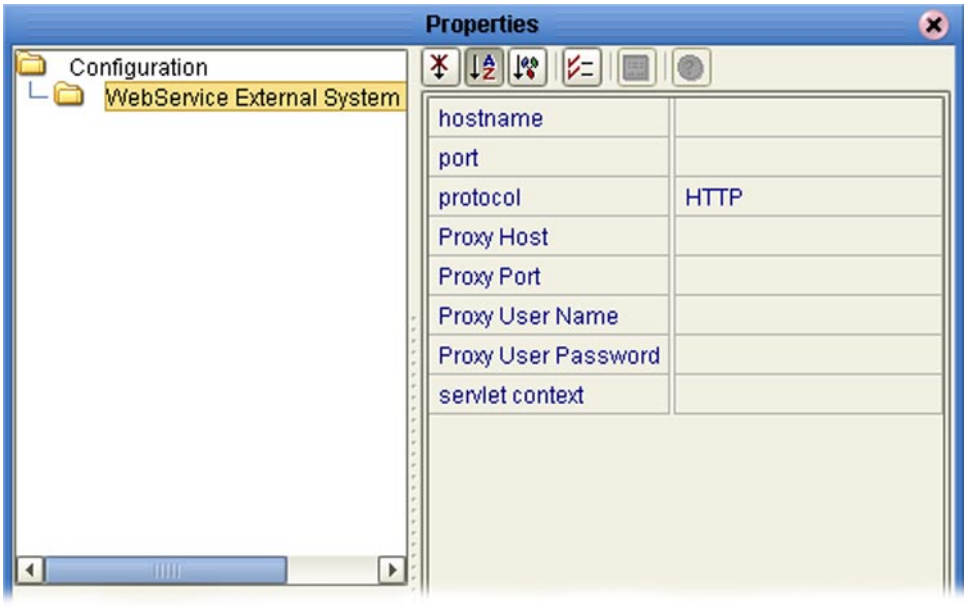


Table 120 Web Services External System Properties

Property	Description
hostname	Name or IP address of the computer hosting the application. There is no default.
port	Port on which the application can be accessed. There is no default.
protocol	WSDL end point protocol (**HTTP** or **HTTPS**). If SSL is enabled for the Web Container (see **Default Web Server** on page 363), this parameter must be set to HTTPS prior to deployment. The default is **HTTP**.
Proxy Host	Name of the proxy host, if any. There is no default.
Proxy Port	Port for the proxy host, if any. There is no default.
Proxy User Name	User name for the proxy host, if any. There is no default.
Proxy User Password	User password for the proxy host, if any. There is no default.
servlet context	The path and name of the Web Services Application.

If you do not specify the properties for the Web Service, then they are assigned automatically upon deployment. The name automatically assigned to the **servlet context** property can be lengthy and complex, however, so you should rename it to something relatively simple. This property controls the name of the Web Service Application (**.war**) file.

Additionally, if you are creating more than one Web Services Application for deployment in any given Environment, you must ensure that the respective servlet context properties have different values so that each application will have a different URL. The URL format is **http://hostname:port/servlet context**.

Note: *You must specify the **hostname and port** if the Project is not deployed on the local system.*

Important: *If you are using the HTTPS protocol, the value of the hostname property must exactly match that used to create the certificates.*

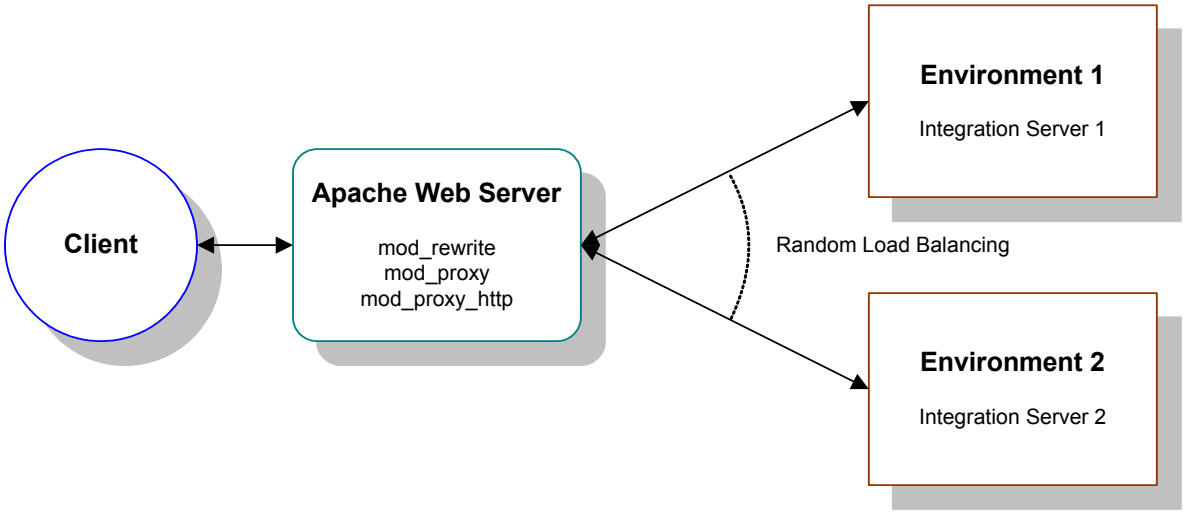
11.6 Load Balancing

You can build a system that will load balance HTTP requests in a way suitable for ICAN Web Services by using an Apache-based Web server. To accomplish this you will need the following:

1. An Apache-based HTTP Web server such as Apache 2.0 or IBM HTTP 2.0.
2. Modify the Apache configuration file to enable **mod_rewrite**, **mod_proxy**, and **mod_proxy_http**.
3. Create a mapping file that includes the name of your Business Process that is published as a Web Service and a list of URL endpoints where it is available.
4. Add the **mod_rewrite** rewrite rule to your Apache configuration file.

Note: *This scheme only provides simple random load-balancing. It does not provide fail-over nor does it provide any heuristics or any other ability to distribute work based on server load.*

Figure 349 Load Balancing Example



11.6.1 Configuring the Apache Server

The main configuration file for Apache, **httpd.conf**, is located in *APACHE*/config. To enable **mod_proxy**, **mod_proxy_http**, and **mod_rewrite**, uncomment the following lines shown in bold text (these options are usually disabled—commented out—by default).

```
#LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule setenvif_module modules/mod_setenvif.so
```

11.6.2 Creating the Mapping File

Before you create the Web Services and the mapping file you need to be aware of the mapping rule that will be used. The URL generated for the ICAN Web Services endpoint is of the following form (you must rename the ***bold_italic*** section in the servlet context property as described in **Web Services External System** on page 418):

```
WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService
```

The mapping file (**ws_server.txt**.) uses the Business Process name as the key to find a list of URL endpoints on which that service will be available. This file is a simple text file of the following form:

```
Key alternate_URL1|alternate_URL2
```

The content below is a typical example—however, it should actually appear on a single line with no space between the alternate URLs. The alternate URLs contain the actual endpoints for the Web Services, including the host IP address and port.

```
bpCalc1
     127.0.0.1:18004/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService|
     127.0.0.1:19004/WSServlet_WSCalc_u02F_IS2_WSC_WSCALC_bpCalc1/STCService
```

The mapping file must be stored on the same system that is running Apache, and is usually found in the configuration directory, for example:

```
C:\Program Files\IBM HTTP Server 2.0\conf\ws_server.txt.
```

11.6.3 Adding the Rewrite Rule

The code section below is the rewrite rule; lines beginning with '**#**' are comments:

```
# Start of ICAN rewrite rules
<IfModule mod_rewrite.c>
#RewriteLog "C:\Program Files\IBM HTTP Server 2.0\logs\rewrite.log"
#RewriteLogLevel 5
RewriteMap SERVICES "rnd:C:\Program Files\IBM HTTP Server
2.0\conf\ws_server.txt"

<Location ~ /WSServlet_ >
    RewriteEngine On

    RewriteRule "([^/_]+(?=/STCService))" "http://${SERVICES:$1}" [P,L]
    RewriteRule "(.*)" "http://localhost:80/notfound.html" [G]
</Location>
</IfModule>
# End of rewrite directives.
```

The purpose of this rule is to intercept requests for ICAN-formatted URLs of the form

```
http://LOADBALANCER:9999/WSServlet_XXXXXXX_bpCalc1/STCService
```

and rewrite them to a form using the entries in the mapping file **ws_server.txt**. It does this by extracting the key (shown in bold) from the requested URL. This key is the last element of the variable part of the endpoint URL created by ICAN.

- If it cannot find a suitable match, it will return a *gone* (410) error.
- If it finds a suitable key but does not find a match in the mapping file, it will return a *not found* (404) error.
- If a key is matched but the service is not available on that URL, it will return a *bad gateway* (502) error.

11.6.4 Debugging

There are a number of ways to debug the internal processing of Apache, the rewrite logging probably being the most useful. You can specify an additional rewrite log and a log level within the rewrite section of the **httpd.conf** file. The logging level is on a scale of 0 to 5, with 0=off and 5=maximum debugging.

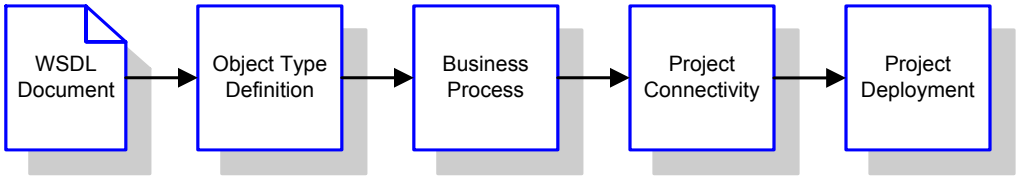
An example of a successful rewrite request log is shown below, using log level 5 (timestamps omitted):

```
[rid#596d60/initial] (3) [per-dir /WSServlet_/] add path info postfix: C:/
     Program Files/IBM HTTP Server 2.0/htdocs/en_US/WSServlet_WSCalc_
     u02F_IS1_WSC_WSCALC_bpCalc1 -> C:/Program Files/IBM HTTP Server 2.0/
     htdocs/en_US/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService

[rid#596d60/initial] (3) [per-dir /WSServlet_/] applying pattern '([^/_]+(?=/
     STCService))' to uri 'C:/Program Files/IBM HTTP Server 2.0/htdocs/en_US/
     WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService'

[rid#596d60/initial] (5) cache lookup OK: map=SERVICES[txt] key=bpCalc1 ->
     val=127.0.0.1:18004/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/
     STCService

[rid#596d60/initial] (5) randomly choosen the subvalue `127.0.0.1:18004/
     WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/STCService'

[rid#596d60/initial] (2) [per-dir /WSServlet_/] rewrite C:/Program Files/IBM
     HTTP Server 2.0/htdocs/en_US/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_
     bpCalc1/STCService -> http://127.0.0.1:18004/WSServlet_WSCalc_u002F_
     IS1_WSC_WSCALC_bpCalc1/STCService

[rid#596d60/initial] (2) [per-dir /WSServlet_/] forcing proxy-throughput with
     http://127.0.0.1:18004/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_bpCalc1/
     STCService

[rid#596d60/initial] (1) [per-dir /WSServlet_/] go-ahead with proxy request
     proxy:http://127.0.0.1:18004/WSServlet_WSCalc_u02F_IS1_WSC_WSCALC_
     bpCalc1/STCService [OK]
```

11.7 Creating a Web Service

A simplified process flow for developing a Web Service is shown in Figure 350.

Figure 350 Development Flow



11.7.1 Procedure

The basic procedure for building a Web Service uses eGate Integrator combined with eInsight Business Process Manager.

To create a Web Service

1. Create the Web Service Project (see **Creating a Project** on page 105).
2. Add a new Object Type Definition (OTD) to the Project.
3. Build the OTD using the WSDL OTD Wizard, based on an externally-provided WSDL file.
4. Add a New Business Process to the Project.
5. Develop the Business Process in eInsight Business Process Manager, based on the OTD.
6. Map the Project components.
7. Deploy the Project to the selected Environment.

11.7.2 Example

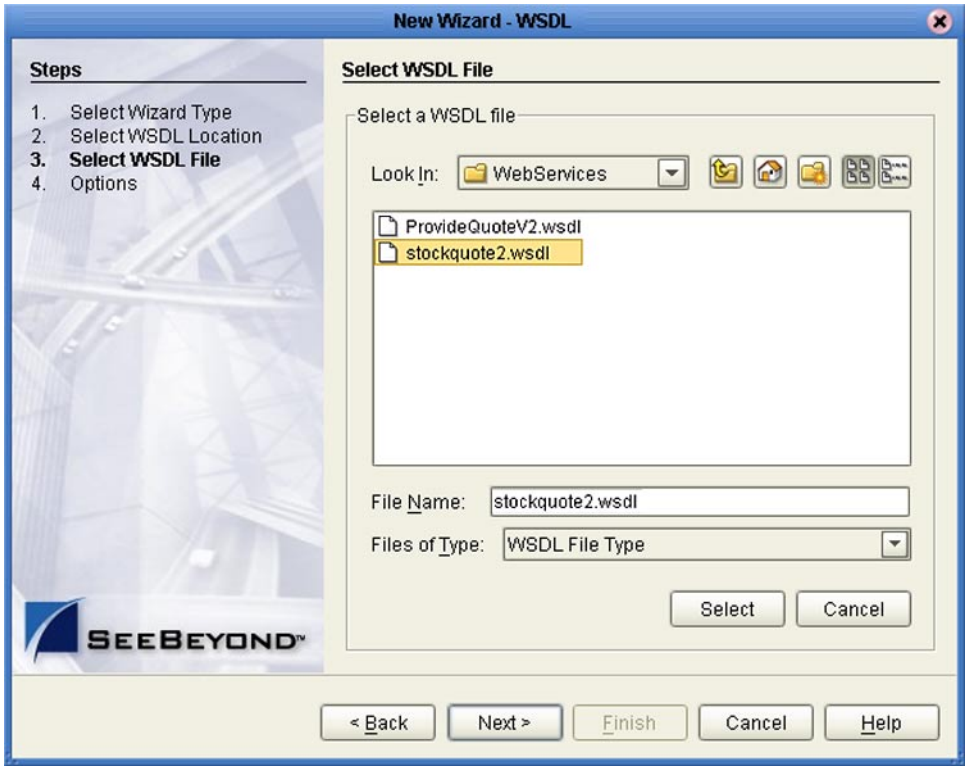
The Project used in the following example is contained in the *Web Services Sample* listed on the Enterprise Manager's Documentation page. To use this example Project, download the sample file **Web_Services_Sample.zip** and extract the contents to a convenient directory. Import the file **webserver.zip** into your Repository following the procedure described in **Project/Environment Import** on page 68.

Note: *For the example Project to work correctly, you must first create and deploy your Web Service, then create and deploy the Web Services Client—deploying both to the same Web Service External System.*

Object Type Definition

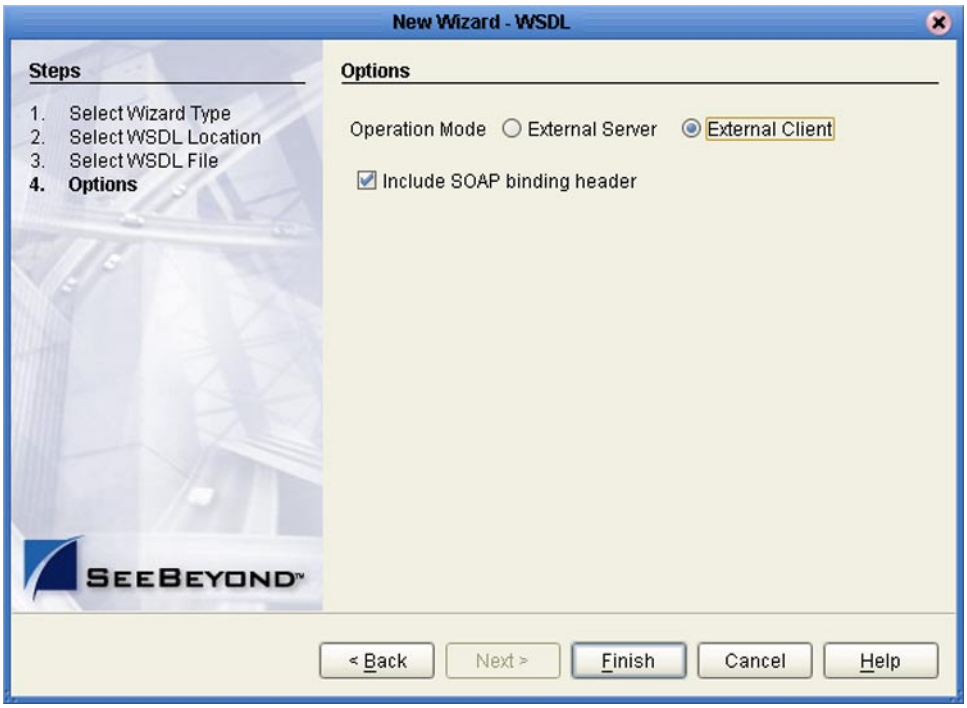
The OTD for the sample Web Service was created as described in **Using the WSDL Wizard** on page 141. The WSDL file must be provided by an external source; the file used in the example is shown in Figure 351.

Figure 351 Wizard Step 3 - Select WSDL File



In the *Options* step, the Operational Mode for the Web Service is **External Client** (see Figure 352).

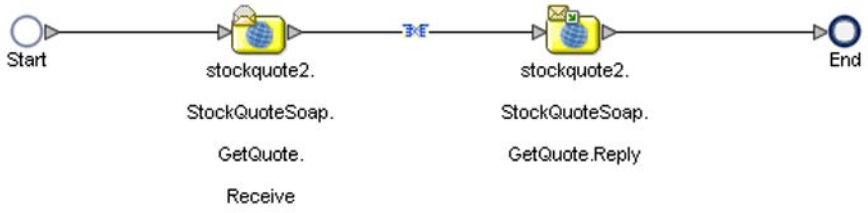
Figure 352 Wizard Step 4 - Options



eInsight Business Process

The example Business Process, developed in eInsight Business Process Manager, is shown in Figure 353 (see the *eInsight Business Process Manager User's Guide* for details).

Figure 353 Web Service Business Process



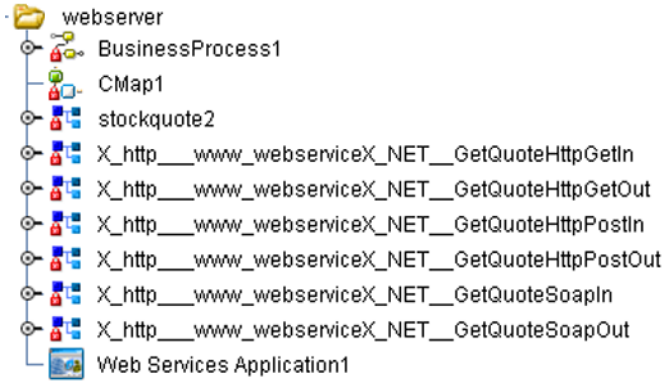
Project Mapping

The Project components are created and connected in the Enterprise Designer Connectivity Map Editor. The example Project contains:

- An External Web Services Application.
- A service, into which you drag and drop the eInsight Business Process from the Project Explorer.

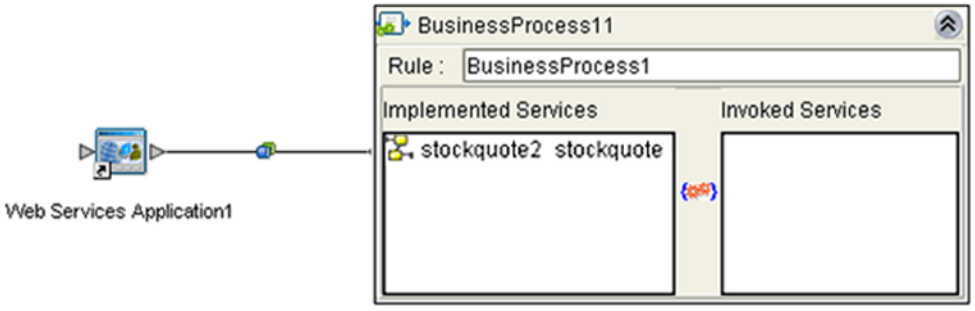
The Web Service example Project appears in the Project Explorer as shown in Figure 354

Figure 354 Web Service Example Project - Explorer Tree



The Business Process is connected as shown in Figure 355.

Figure 355 Web Service Example Project - Business Process Mapping



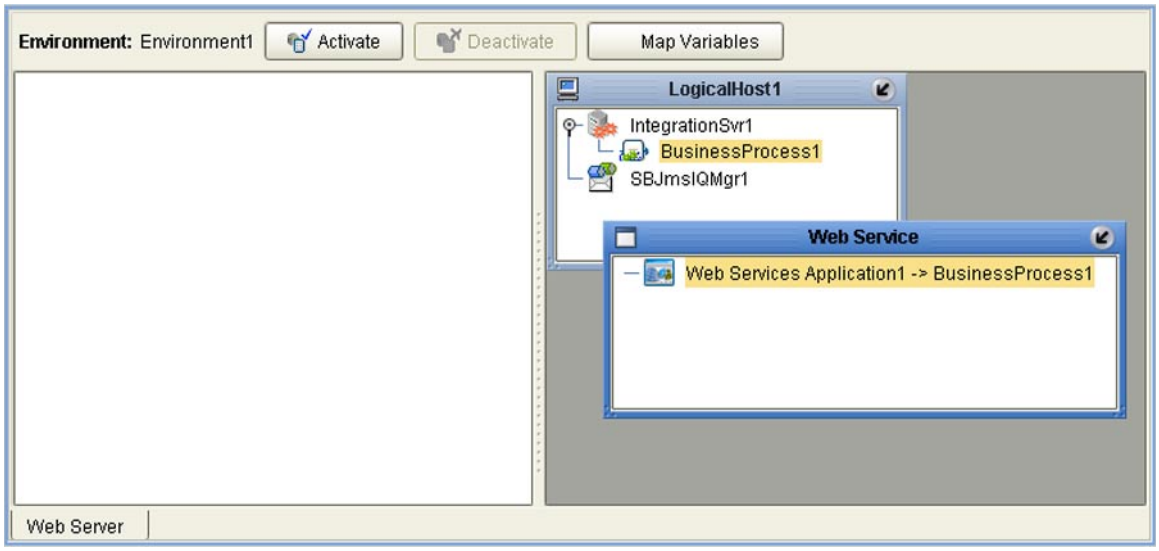
Collapsing the binding box results in Figure 356.

Figure 356 Web Service Example Project - Connectivity Map

Project Deployment

The Project is deployed to a run-time Environment as shown in Figure 357. (See **Creating a Deployment Profile** on page 397 and **Activating and Deactivating Deployment Profiles** on page 403 for information.)

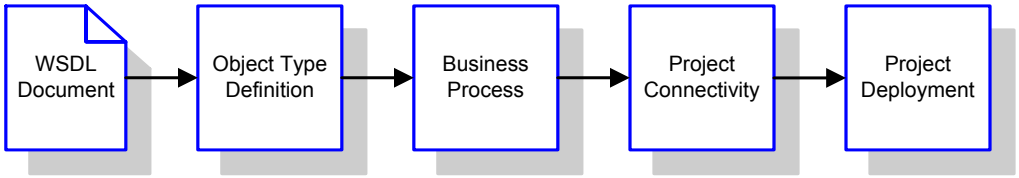
Figure 357 Web Service Example Project - Deployment



11.8 Building a Web Services Client

A simplified process flow for developing a Web Services Client is shown in Figure 358.

Figure 358 Development Flow



11.8.1 Procedure

As with a Web Service, the basic procedure for building a Web Services Client uses eGate Integrator combined with eInsight Business Process Manager.

To build a Web Services Client

1. Create the Web Services Client Project (see **Creating a Project** on page 105).
2. Add a new Object Type Definition (OTD) to the Project.
3. Build the OTD using the WSDL OTD Wizard, based on an externally-provided WSDL file.
4. Add a New Business Process to the Project.
5. Develop the Business Process in eInsight Business Process Manager, based on the OTD.
6. Map the Project components.
7. Deploy the Project to the selected Environment.

11.8.2 Example

The Project used in the following example is contained in the *Web Services Sample* listed on the Enterprise Manager's Documentation page. To use this example Project, download the sample file **Web_Services_Sample.zip** and extract the contents to a convenient directory. Import the file **webclient.zip** into your Repository following the procedure described in **Project/Environment Import** on page 68. The files **input.txt** and **output.txt** are simple text files for testing purposes.

Note: *For the example Project to work correctly, you must first create and deploy your Web Service, then create and deploy the Web Services Client—deploying both to the same Web Service External System.*

Object Type Definition

The OTD for the sample Web Services Client was created as described in **Using the WSDL Wizard** on page 141. The required WSDL file (see Figure 359) is produced by the Web Service developed in the preceding section, and published in the UDDI registry upon deployment and activation.

Figure 359 Example Web Service WSDL File

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions targetNamespace="http://seebeyond/quoteservice" xmlns:tns="http://seebeyond/quoteservice"
    xmlns:sbynpx="http://bpel.seebeyond.com/hawaii/5.0/privateExtension/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  - <types targetNamespace="http://seebeyond/quoteservice">
    - <xsd:schema targetNamespace="http://seebeyond/quoteservice"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:element type="xsd:string" name="QuoteRequest" />
        <xsd:element type="xsd:string" name="QuotePrice" />
        <xsd:element type="xsd:string" name="QuoteCompany" />
      </xsd:schema>
    </types>
  - <message name="msgProvideQuoteRequest">
      <part name="TICKER" element="tns:QuoteRequest" />
    </message>
  - <message name="msgQuoteResponse">
      <part name="price" element="tns:QuotePrice" />
      <part name="company" element="tns:QuoteCompany" />
    </message>
  - <portType name="ptProvideQuote">
```

The file location is most easily specified by locating the file using the UDDI browser (see Figure 360), and entering its URL in the WSDL Wizard as shown in Figure 361.

Figure 360 SeeBeyond UDDI Browser

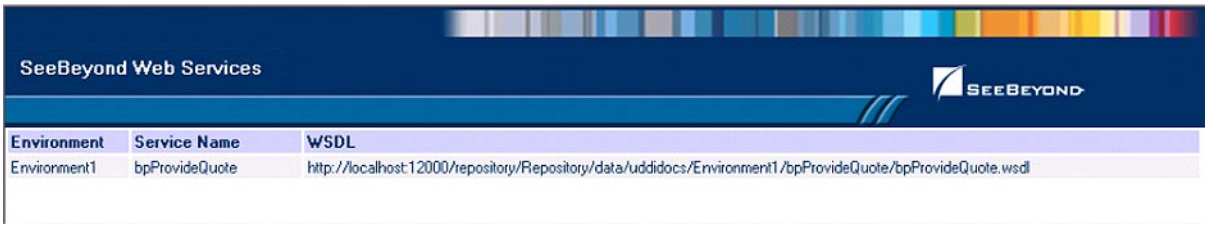
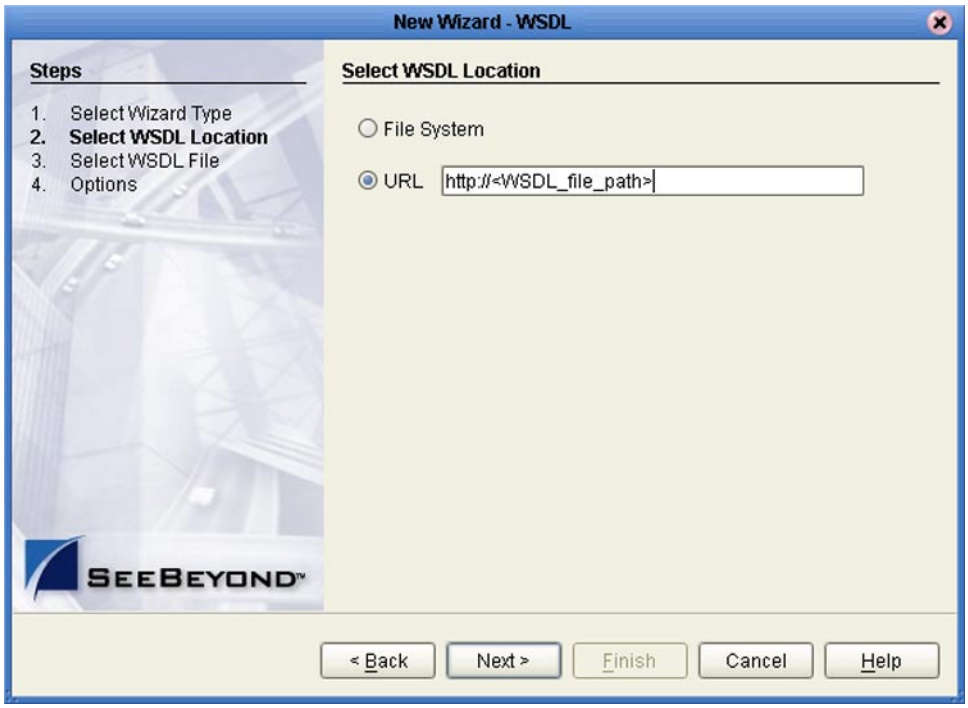
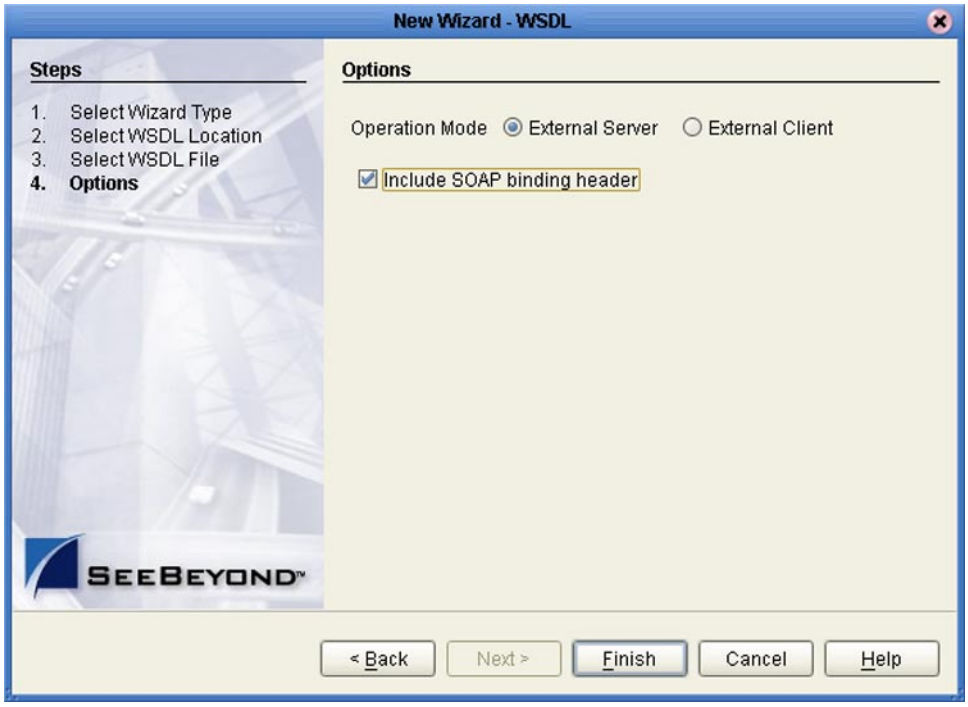


Figure 361 Select WSDL File



In the *Options* Wizard, the Operational Mode for the Web Services Client is **External Server** (see Figure 362).

Figure 362 Wizard Step 4 - Options



eInsight Business Process

The example Business Process, developed in eInsight Business Process Manager, is shown in Figure 363 (see the *eInsight Business Process Manager User's Guide* for details).

Figure 363 Web Services Client Business Process

The **receive** rule for the Business Process is shown in Figure 364, and the **write** rule is shown in Figure 365.

Figure 364 Web Services Client Business Process *Receive* Rule

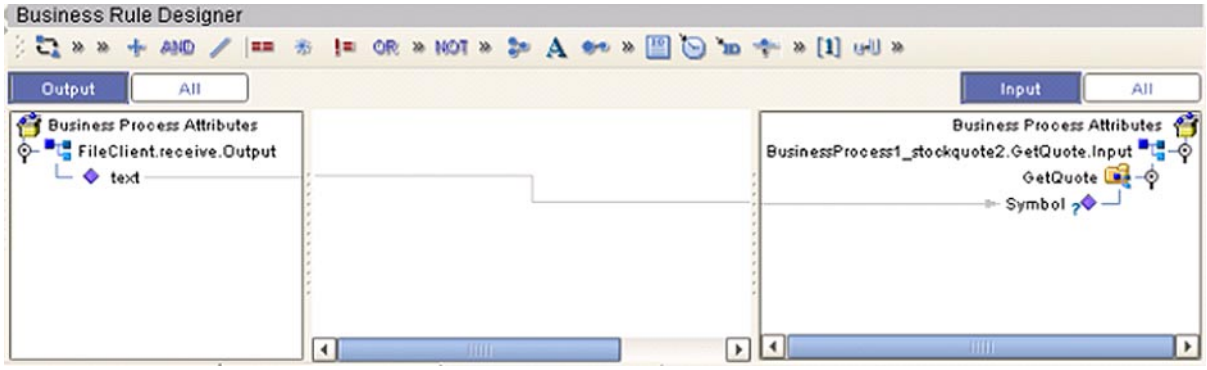
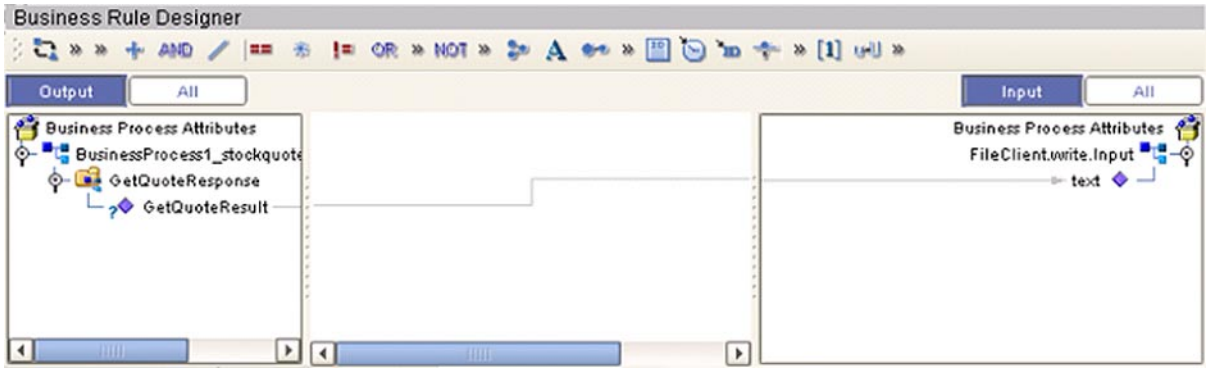


Figure 365 Web Services Client Business Process *Write* Rule



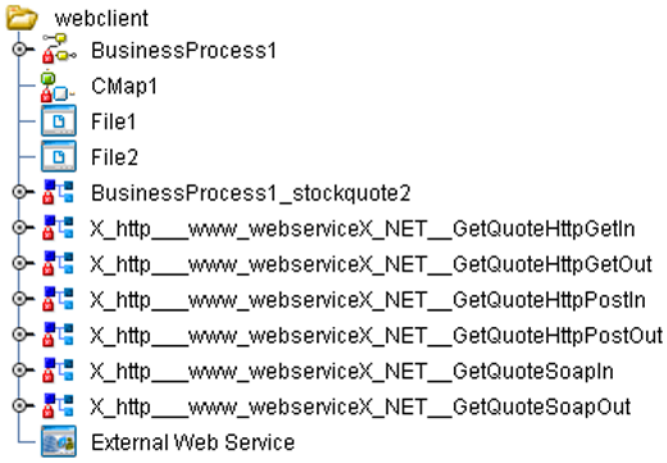
Project Mapping

The Project components are created and mapped in the Enterprise Designer Connectivity Map Editor. The example Project contains:

- Two external files and accompanying File eWays.
- An External Web Service.
- A service, into which you drag and drop the eInsight Business Process from the Project Explorer.

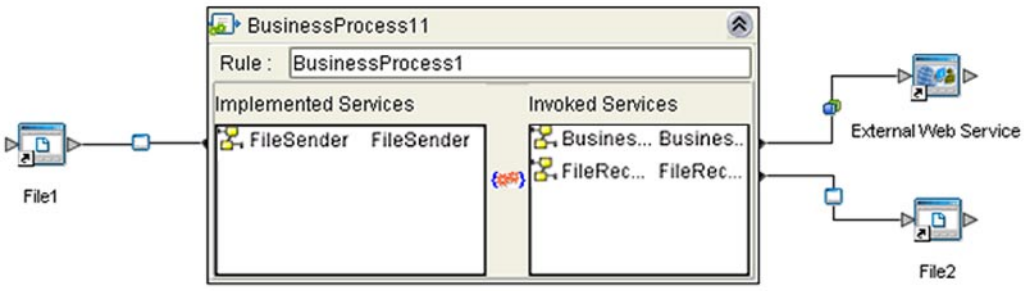
The Web Services Client example Project appears in the Project Explorer as shown in Figure 366.

Figure 366 Web Services Client Example Project - Explorer Tree



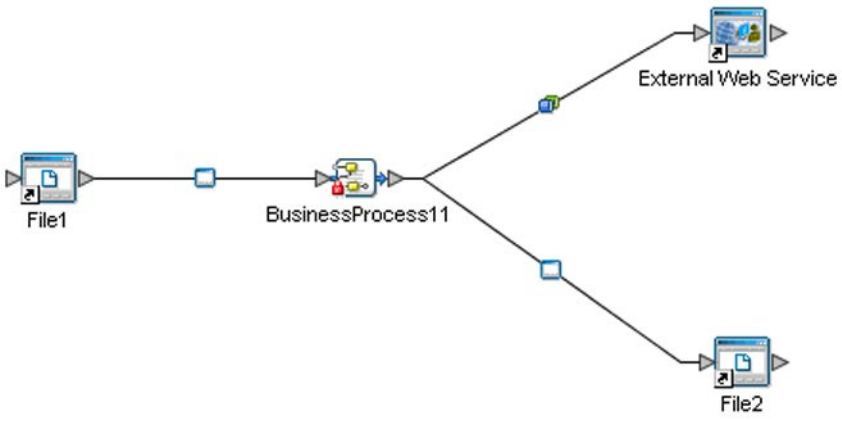
The Business Process is connected as shown in Figure 367.

Figure 367 Web Services Client Example Project - Business Process Mapping



Collapsing the binding box results in Figure 368.

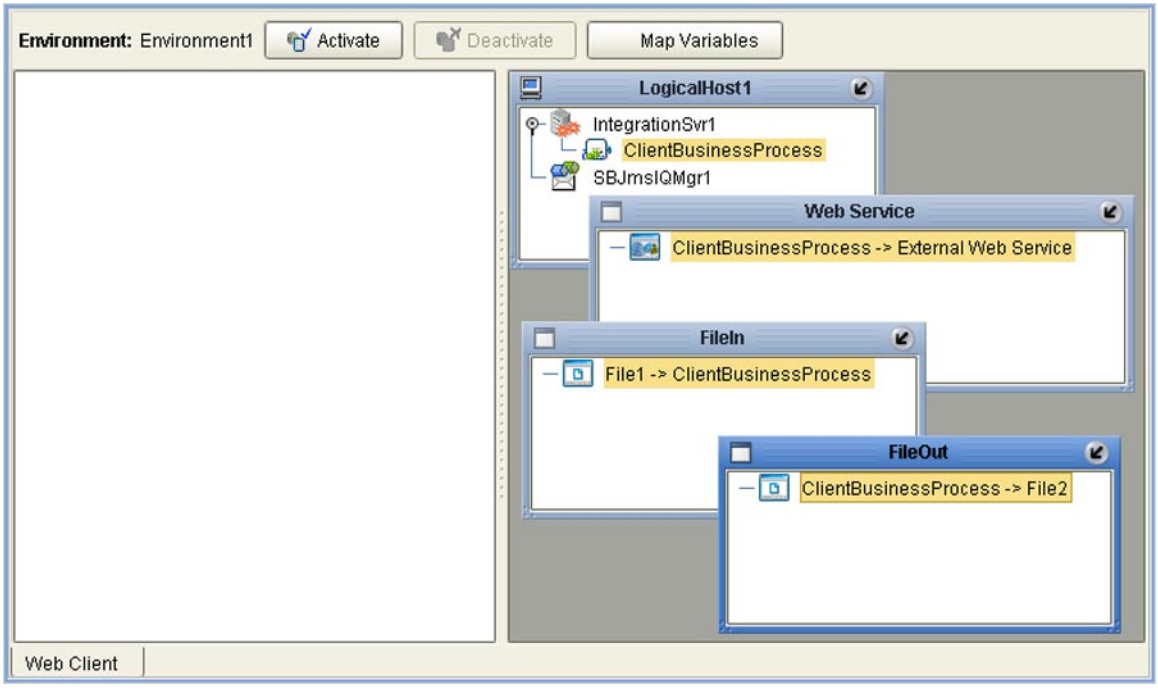
Figure 368 Web Services Client Example Project - Connectivity Map



Project Deployment

The example Project is deployed to a run-time Environment as shown in Figure 369. (See **Creating a Deployment Profile** on page 397 and **Activating and Deactivating Deployment Profiles** on page 403 for information.)

Figure 369 Web Services Client Example Project - Deployment



If the Web Service and Web Services Client are deployed to different Web Services External Systems (usually in different Environments), you must manually copy the configuration from the Web Services External System in which the Service is deployed to the one in which the Client is deployed. See **Web Services External System** on page 418.

Project Operation

To run the Web Services Example Project

1. Start the Logical Host (see **Bootstrapping** on page 356).
2. Copy the file **input.txt** into the directory that the **File1** eWay polls for input files (**C:\temp**).
3. A valid output file (see **output.txt**) should appear in the directory into which the **File2** eWay places files (**C:\temp**).

11.9 Setting Up Secure Web Services

11.9.1 SSL Overview

Secure Sockets Layer (SSL) allows Web browsers and Web servers to communicate over a secured connection. In this secure connection, the data that is being sent is *encrypted* before being sent, and *decrypted* upon receipt and prior to processing. Both the browser and the server encrypt all traffic before sending any data.

Another important aspect of the SSL protocol is *authentication*. During your initial attempt to communicate with a Web server over a secure connection, the server will present your Web browser with a set of credentials in the form of a server certificate. The purpose of the certificate is to verify that the site is who and what it claims to be. In some cases, the server may request a certificate to verify that the client is who and what it claims to be. This is known as client authentication.

Certificates and Keys

In order to implement SSL, a Web server must have an associated certificate for each external interface, or IP address, that accepts secure connections. The theory behind this design is that a server should provide some kind of reasonable assurance that its owner is who you think it is, particularly before receiving any sensitive information. It may be useful to think of a certificate as a "digital driver's license" for an Internet address. It states with which company the site is associated, along with some basic contact information about the site owner or administrator.

A certificate is a digitally signed statement from one entity (person, company, and so on), indicating that the public key (and some other information) of some other entity has a particular value. When data is digitally signed, the signature can be verified to check the data integrity and authenticity. *Integrity* means that the data has not been modified or tampered with, and *authenticity* means that the data indeed comes from whoever claims to have created and signed it.

The certificate is cryptographically signed by its originator and is difficult for anyone else to forge. For sites involved in e-commerce, or any other business transaction in which authentication of identity is important, a certificate can be purchased from a well-known Certificate Authority (CA) such as Verisign or Thawte. For testing purposes or use within an adequately controlled environment, a self-signed certificate can be used instead.

Certificates are used with the HTTPS protocol to authenticate Web servers to Web clients, and vice versa. The HTTPS service of the ICAN Repository server will not run unless a server certificate has been installed. Use the following procedure to set up a server certificate that can be used by the ICAN Repository server to enable SSL.

Keytool Utility

One tool that can be used to set up a server or client certificate is **Keytool**, a key and certificate management utility that ships with the J2SE SDK. It enables users to administer their own public/private key pairs and associated certificates for use in self-authentication (where the user authenticates himself or herself to other users or services) or data integrity and authentication services, using digital signatures. It also allows users to cache the public keys (in the form of certificates) of their communicating peers.

The keys and certificates are stored in a *keystore*. The default keystore implementation implements the keystore as a file. It protects private keys with a password.

The **Keytool** utility enables you to create the certificate. The version that ships with the J2SE SDK programmatically adds a Java Cryptographic Extension provider that has implementations of RSA algorithms. This provider enables you to import RSA-signed certificates.

Note: *If the Web Service and Web Services Client are deployed to different Web Services External Systems (usually in different Environments), you must manually copy the configuration from the Web Services External System in which the Service is deployed to the one in which the client is deployed. See* **Web Services External System** *on page 418.*

11.9.2 Web Services Setup

Setup procedures for using SSL are described in detail in **Configuring SSL Support** in the *eGate Integrator System Administration Guide*. A brief outline of these procedures is included in the following.

Note: *You must first create and deploy your Web Service, then create and deploy the Web Services Client—deploying both to the same Web Service External System.*

Web Service Setup

To set up SSL for a Web Service

1. Create the Web Service as described in **Creating a Web Service** on page 423.
2. Configure your Integration Server to use SSL.
   - A Configure the Web Container (see **Web Container Configuration** on page 362 and **Default Web Server** on page 363). If you have already deployed non-SSL Web Services Projects using properties defined in *Web Container Configuration: Default Web Server*, then you must create a new set of configuration properties as described in **Web Server Configurations** on page 362, enabling SSL and using a different Connector Port.
   - B Create a new Web Services External System (see **Web Services External System** on page 418), setting the **protocol** property to **HTTPS**. The **port** number must match the **Connector Port** in the corresponding Web Server, as set in step A above.

Note: *If you need a self-signed Certificate, skip step C and create the server Certificate and Keystore using the script in* **Creating Keystores and Self-Signed Certificates** *on page 445. Then continue with step D.*

   - C Right-click the Environment and create a new **Keystore**.
   - D Right-click the Keystore and select **Manage Trust Stores**.
   - E Click **New** and give the Truststore an alias name. The Certificate fields will be enabled.
   - F Import the server Certificate into the Truststore.
3. Create a new Deployment Profile (see **Creating a Deployment Profile** on page 397).
4. Start the Logical Host (see **Bootstrapping** on page 356), which creates a new **keystore** directory under ***ICAN-root*\logicalhost**.
5. Copy the file **IntegrationSvr1.keystore** into this directory, which will allow the Web Container to find the SSL Certificate.
6. Activate the Web Service Project (see **Activating and Deactivating Deployment Profiles** on page 403), which should restart the Logical Host.

Web Services Client Setup

To set up SSL for a Web Services Client

1. Create the Web Services Client as described in **Building a Web Services Client** on page 428.
2. Import the server Certificate into the client Truststore using the command line:
   A. For design time, use the command:
      Keytool -keystore cacerts -alias ***aliasname*** -import - file server.cer
      ICAN-root(client)\edesigner\jdk\jre\lib\security
   B. For run time, use the command:
      Keytool -keystore cacerts -alias ***aliasname*** -import - file server.cer
      ICAN-root(client)\logicalhost\jre\lib\security
3. Create a new Deployment Profile, deploying the Web Services Client to the same Web Services External System as the Web Service (see **Creating a Deployment Profile** on page 397).
4. Activate the Project (see **Activating and Deactivating Deployment Profiles** on page 403).
5. If the Logical Host is running, restart it (see **Bootstrapping** on page 356).

11.9.3 Web Services Example Project

An example Web Services Project is contained in the *Web Services SSL Sample* listed on the Enterprise Manager's Documentation page. To use this example Project, download the sample file **Web_Services_SSL_Sample.zip** and extract the contents to a convenient directory.

To make the example Project self-contained, self-signed Certificates are used for authentication. A script is provided in a batch file to generate the required Keystores and Certificates, and import the Certificates into the appropriate Keystores. The example does not incorporate encryption.

Note: *For the example Project to work correctly, you must first create and deploy your Web Service, then create and deploy the Web Services Client—deploying both to the same Web Service External System.*

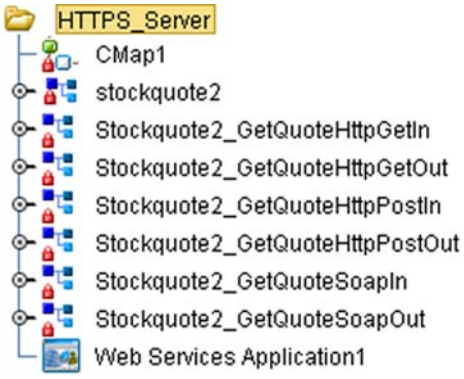
Web Service Example

Note: *You must have eInsight installed in your ICAN system.*

To implement the example Web Service

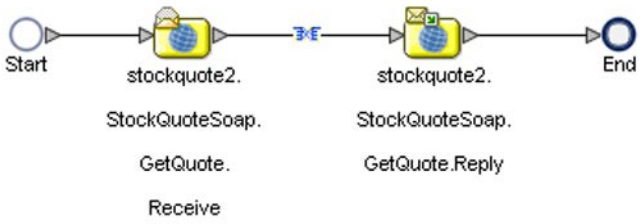
1 Import the file **HTTPS_Server.zip** into your Repository following the procedure described in **Project/Environment Import** on page 68. The Project should appear in the Project Explorer as shown in Figure 370.

Figure 370 Example Secure Web Services Project - Explorer Tree



2 The Business Process should appear as shown in Figure 371.

Figure 371 Example Secure Web Services Project - Business Process



3 The Project Connectivity Map should appear as shown in Figure 372.

Figure 372 Example Secure Web Services Project - Connectivity Map

4 Configure the Default Web Server to use SSL (see **Default Web Server** on page 363).

A *Enable SSL:* **true**

B *SSL Keystore Password:* **tomcat**

C *Connector Port: (default)*

Note: *If you have already deployed non-SSL Web Services Projects using properties defined in Web Container Configuration: Default Web Server, then you must create a new set of configuration properties as described in* **Web Server Configurations** *on page 362, enabling SSL and using a different Connector Port.*

5 Create a new Web Services External System (see **Web Services External System** on page 418).

A *hostname:* the IP address or domain name where the Logical Host will be running.

B *port:* the same as the **Connector Port** specified in step 4 above.

C *protocol:* **HTTPS**

6 Create the server Keystore and the required self-signed certificates (see **Creating Keystores and Self-Signed Certificates** on page 445).

7 Right-click the Keystore and select **Manage Trust Stores**.

8 Click **New** and give the store an alias name. The *Certificate* fields will be enabled.

9 Import the server Certificate into the Truststore.

10 Create a new Deployment Profile (see **Creating a Deployment Profile** on page 397).

11 Start the Logical Host (see **Bootstrapping** on page 356), which creates a new **keystore** directory under ***ICAN-root*\logicalhost**.

12 Copy your **IntegrationSvr1.keystore** file into this directory, which will allow the Web Container to find the SSL certificate.

13 Activate the Web Service Project (see **Activating and Deactivating Deployment Profiles** on page 403), which should restart the Logical Host.

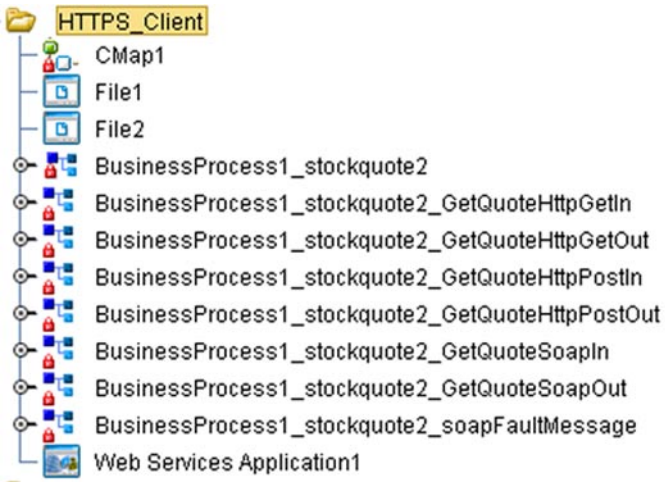
Web Services Client Example

Note: *You must have eInsight installed in your ICAN system.*

To implement the example Web Services Client

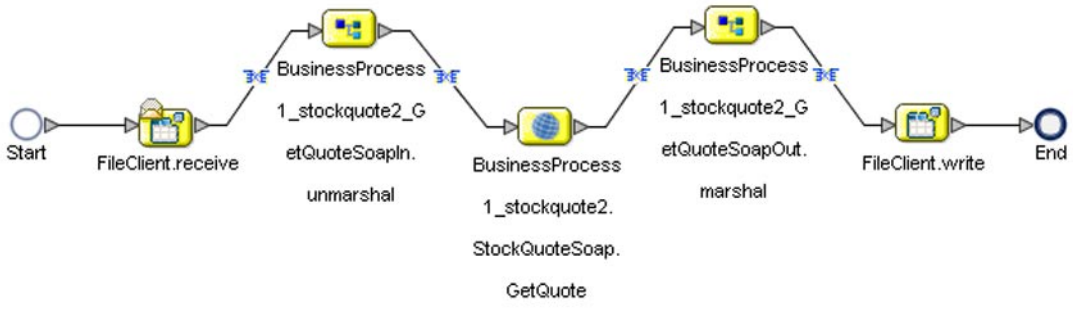
1 Import the file **HTTPS_Client.zip** into your Repository following the procedure described in **Project/Environment Import** on page 68. The Project should appear in the Project Explorer as shown in Figure 373.

Figure 373 Example Secure Web Services Client Project - Explorer Tree



2 The Business Process should appear as shown in Figure 374.

Figure 374 Example Secure Web Services Client Project - Business Process



3 The Project Connectivity Map should appear as shown in Figure 375 and Figure 376.

Figure 375 Example Secure Web Services Client Project - Business Process Mapping

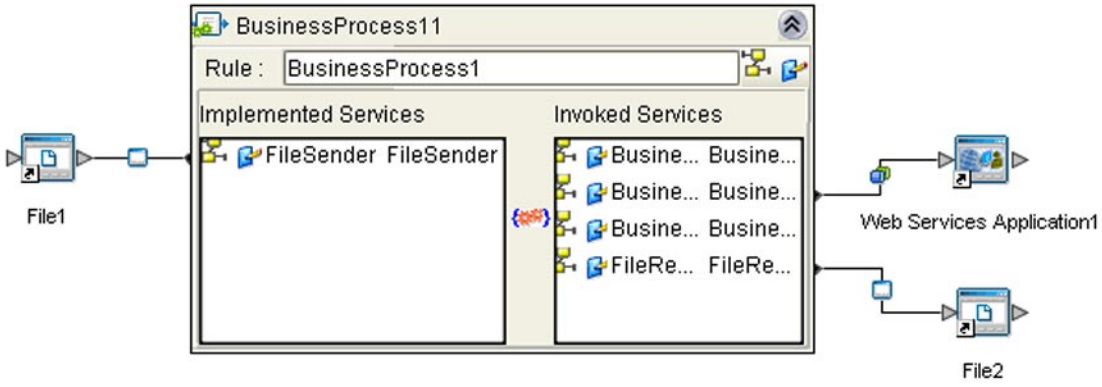
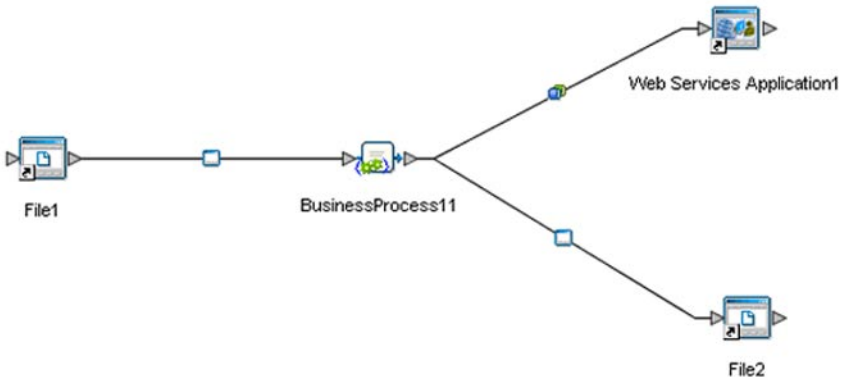


Figure 376 Example Secure Web Services Client Project - Connectivity Map



4 Open a Command Prompt window and import the server Certificate (server.cer) into the client Truststore (tomcat-cl).

A For design time, use the command:
Keytool -keystore cacerts -alias tomcat-cl -import - file server.cer
ICAN-root(client)\edesigner\jdk\jre\lib\security

B For run time, use the command:
Keytool -keystore cacerts -alias tomcat-cl -import - file server.cer
ICAN-root(client)\logicalhost\jre\lib\security

Note: *When asked, use the default password* ***changeit****.*

5 Create a new Deployment Profile, deploying the Web Services Client to the same Web Services External System as the Web Service (see **Creating a Deployment Profile** on page 397).

6 Activate the Web Services Client Project (see **Activating and Deactivating Deployment Profiles** on page 403).

7 If the Logical Host is running, restart it (see **Bootstrapping** on page 356).

Project Operation

To run the Web Services Example Project

1 Start the Logical Host, if it is not already running (see **Bootstrapping** on page 356).

2 Copy the file **inputssl.~in** into the directory that the **File1** eWay polls for input files (**C:\temp**).

3 A valid output file (see **outputssl1.dat**) should appear in the directory into which the **File2** eWay places files (**C:\temp**).

11.9.4 Creating Keystores and Self-Signed Certificates

The sample file **Web_Services_SSL_Sample.zip** contains a batch file (**key_cert.bat**) containing the following script for use with the example Project. This script will:

- Generate the server and client Keystores.
- Create a self-signed certificate for the server.

The parameter **<servername>** must be a fully-qualified domain name, and must match the **hostname** in the Web Services External System configuration. The default password for the certificates is **tomcat**, which you can change if desired.

You should set the environmental variable **JAVA_HOME** to point to your JDK installation (***ICAN_root*\repository\jre\1.4.2_04**).

Example Project Script

```
if not "%JAVA_HOME%" == "" goto gotJavaHome
goto cleanup
:gotJavaHome

echo Generating the Server Keystore in file IntegrationSvr1.keystore
%java_home%\bin\keytool -genkey -alias tomcat-sv -dname
    "CN=<servername>, OU=X, O=Y, L=Z, S=XY, C=YZ" -keyalg RSA -
    keypass tomcat -storepass tomcat -keystore
    IntegrationSvr1.keystore

echo Exporting the Server's Certificate from Server's Keystore to an
    external file server.cer
%java_home%\bin\keytool -export -alias tomcat-sv -storepass tomcat -
    file server.cer -keystore IntegrationSvr1.keystore

echo Generating the Client Keystore in file client.keystore
%java_home%\bin\keytool -genkey -alias tomcat-cl -dname
    "CN=<servername>, OU=X, O=Y, L=Z, S=XY, C=YZ" -keyalg RSA -
    keypass tomcat -storepass tomcat -keystore client.keystore

:cleanup
```

Chapter 12

Troubleshooting

This chapter contains descriptions of potential solutions for problems you may encounter when using eGate Integrator.

What's in This Chapter

12.1 Error Messages

12.1.1 OutOfMemory

Out-of-memory errors occurring during Project design are most frequently overcome by increasing the *heap size* of the editor you are using, or of Enterprise Designer itself (see **Options Setup** on page 57). Try increasing the heap size in increments of 50 MB, starting with the appropriate editor.

If you are using an XSD OTD having a file size in excess of 1 MB, and experience out-of-memory errors that cannot be overcome by increasing the heap size, you may need to increase the *permanent generation* memory size. To do so, you must edit the **runed.bat** file (found in ***ICAN-root*\edesigner\bin**), adding the following command-line parameters after the command **runidew** (near the end of the file):

```
-J-XX:PermSize=192m -J-XX:MaxNewSize=128m -J-XX:MaxPermSize=192m
```

Glossary

BPEL
BPEL (Business Process Execution Language), also known as BPEL4WS (Business Process Execution Language for Web Services), is an XML-based language designed to enable task sharing for either a distributed or grid computing environment. It combines and replaces **WSDL** and Microsoft's XLANG specification.

Collaboration
A logical operation performed between some combination of message destinations and external applications. The operation is defined by a Collaboration Definition (see next entry), which can be encoded in either Java or **XSLT**.

Collaboration Definition
The encoding of business rules, in Java or XSLT format. Typically, the encoding consists of operations on an **Object Type Definition (OTD)**. Several Collaborations can have the same Collaboration Definition.

Connection
Consists of the configuration information that enables an eWay to connect to an external system.

Connectivity Map
Contains business logic and routing information about the data transmission. A Connectivity Map usually includes one or more **Collaboration**s, **Topic**s, **Queue**s, and **eWay**s. A Connectivity Map is created under a **Project**. A Project may have multiple Connectivity Maps.

Constant
A static name-value pair that is visible across a **Project**.

CRM
Acronym for Customer Relations Management.

Data Cleansing
Data must be “cleansed” of errors in structure and content for accurate and effective use in a database or data management systems.

Data Dictionary
Defines the organization of a database and lists all files in the database, the number of records in each file, and the names and types of each field. Although the dictionary does not contain actual data, it does contain essential information for managing the database. The data dictionary is often hidden from end users

Data Integrity
Refers to the accuracy and validity of data. Data integrity can be compromised in many ways, including human error through data entry, or through faulty logic in programming. Computer viruses, software bugs and many other factors can also compromise data integrity.

Data Mapping
Refers to establishing the relationship and data flow pattern between source and target objects, usually within the context of relational database management systems (RDBMSs).

Data Transformation
Data transformation is necessary after extracting data from legacy data formats, or any format that requires **Data Cleansing**. Data is transformed for efficient use for Business-to-Business Enterprise Data Integration.

DBCS
Acronym for Double-Byte Character Set.

Deployment Profile
Contains the information about how the **Project** components will be deployed in an Environment. A Project can have multiple Deployment Profiles, but only one Deployment Profile can be activated for a Project in any one **Environment**.

Derived Collaboration
A Collaboration that inherits operations from another, according to standard object-oriented practice.

DTD
A Document Type Definition (DTD) specifies how an associated document, written either in the Standard Generalized Markup Language (SGML) or of the Extensible Markup Language (**XML**), is to be processed.

Enterprise Designer
The **Project** design tool within eGate Integrator.

Enterprise Service Bus (ESB)
A category of software, incorporating native Web Services support, that provides a low-end alternative to a comprehensive integration broker suite—offering limited functionality, but less complexity and lower cost.

Environment
A collection of physical resources and their configurations that are used to host eGate Integrator **Project** components. An Environment contains Logical Hosts and external systems.

ERM
Acronym for Enterprise Resource Management.

ETL
A three-phase (extract/transform/load) process used, for example, to generate and maintain a central **Metadata** repository.

- **Extract** is the process of reading data from a source database and obtaining the desired subset of data.
- **Transform** is the process of converting the extracted data from its previous form into the desired form.
- **Load** is the process of writing the transformed data into the target database.

eWay
A link between a **Collaboration** and an external connection including the message server connection (topic or queue) or external application.

External Application
A logical representation of an application external to the ICAN Suite.

External System
A representation of a computer system hosting an application external to the ICAN Suite.

Extraction
The process of reading data from a source database and obtaining the desired subset of data (see **ETL**).

HTML
HTML (HyperText Markup Language) is the set of markup symbols or codes (tags) inserted in a file intended for display on a Web page. HTML describes the content of the Web page (primarily text and graphics) only in terms of how it is to be displayed and interacted with.

HTTP
HTTP (HyperText Transfer Protocol) is the set of rules for transferring files—text, graphics, audio, video—on the World Wide Web.

ICAN Suite
The SeeBeyond Integrated Composite Application Network Suite.

Impact Analyzer
A module within Enterprise Designer that analyzes and predicts the impact a specified change would have on other components in the **Project**.

Integration Server
J2EE-compatible software platform that houses the business logic container used to run Collaborations and JCA connectors (eWays). Provides transaction services, persistence, and external connectivity.

JMS IQ Manager
JMS-compliant, guaranteed delivery store, forwarding, and queueing service.

Link
The JMS Connection between a **Collaboration** and a topic or queue in a JMS-compliant message server.

Linked Message Destination
A reference to a **Message Destination** defined in another **Connectivity Map**.

Logical Host
A Logical Host contains the eGate Integrator run-time components, including integration servers and message servers, that are installed on a host hardware platform.

Management Agent
Uses J2EE technology to manage and monitor an eGate Integrator deployment that may contain other application servers in addition to the SeeBeyond Integration Server. Defines management interfaces and services designed for distributed environments, focusing on providing functionality for managing networks, systems, and applications.

Message Destination
A general term for a topic or queue. Two or more Projects can share a message destination that has the same name and is deployed on the same message server. A single Project may also have a single message destination referenced in multiple **Connectivity Map**s.

Metadata
Metadata describes the structure and format of a particular set of data. **ETL** tools are used to generate and maintain a central metadata repository.

Non-normalized Data
Data that has not been converted to normalized data (see next entry).

Normalized Data
Data that has been processed to remove redundant or incorrect data structure and organization, thereby creating a maintainable data set that can be cross-referenced. Normalized data is not only easier to analyze but also easier to expand.

Object Type Definition (OTD)
Object Type Definitions contain the data structure and rules that define an object. OTDs are used in **Collaboration Definition**s for creating data transformations and interfacing with external systems.

Project
Contains a collection of logical components, configurations, and files that are used to solve business problems. A Project organizes the files and packages and maintains the settings that comprise an eGate Integrator system in SeeBeyond's **Enterprise Designer**.

Query
A request for information from a database. There are three basic query methods:

- **Choose** – With this simplified method, the database system presents a list of parameters from which you can choose. This method is the least flexible of the three methods.
- **Query by example (QBE)** – With this method, the system lets you specify fields and values to define a query.
- **Query language** – With this method, you have the ability to make requests for information in the form of a stylized query using a query language. This is the most complex and powerful of the three methods.

Queue

A JMS queue is a shareable object that conforms to the *point-to-point* (p2p, or PTP) messaging domain, where one sender delivers a message to exactly one receiver. When the SeeBeyond **JMS IQ Manager** sends a message to a queue, it ensures it is received once and only once, although there may be many receivers "listening" to the queue. This is equivalent to the subscriber pooling in other queue implementations. You can reference a queue that exists in another **Connectivity Map** or **Project**.

Raw Data

Unprocessed data, as obtained from the source (see also **Non-normalized Data**).

Relational Database (RDBMS)

Short for Relational Database Management System, most often referred to as RDBMS, in which data is stored in related tables and can be viewed in many different ways. Relational databases differ from flat-file databases, in which each database is self-contained as a single file or table.

Repository

Stores and manages the setup, component, and configuration information for eGate Integrator **Project**s. The Repository also provides monitoring services for Projects, which include version control and impact analysis.

SBCS

Acronym for Single-Byte Character Set.

Schema Runtime Environment (SRE)

An add-on feature in eGate Integrator 5.0 that allows **Collaboration**s developed in e*Gate 4.x to be used in and controlled from eGate Integrator 5.0, thereby providing an interim upgrade path for e*Gate 4.x users.

Service

Contains the information about executing a set of business rules. These business rules can be defined in a Java or XSLT **Collaboration Definition**, Business Process, eTL Definition, or other service. A Service also contains binding information for connecting to JMS **Topic**s, **Queue**s, **eWay**s, and other services.

SI/SO

Acronym for Shift-In/Shift-Out.

SOAP

SOAP (Simple Object Access Protocol) enables a program running in one operating system to communicate with another program running in either the same or a different operating system, using **HTTP** and **XML** as the mechanisms for information exchange.

Subproject

An independent **Project** that is included as part of another Project, and is displayed in the Enterprise Explorer tree as a "branch" beneath the main Project.

Table

Refers to data arranged in rows and columns, as in a spreadsheet. In **Relational Database (RDBMS)** systems, all information is stored in tables.

Topic

A JMS topic is a shareable object that conforms to the *publish-and-subscribe* (pub/sub) messaging domain, where one publisher broadcasts messages to one or more subscribers. When the SeeBeyond **JMS IQ Manager** publishes a message on a topic, it ensures that all subscribers receive the message.

Transformation

The process of converting the data extracted from its source into the form required by its target program or system. This process includes **Data Cleansing**, **Data Mapping**, data normalization (see **Normalized Data**), and other sub-processes.

UDDI

UDDI (Universal Description, Discovery, and Integration) is an **WSDL**-based registry that enables businesses to list themselves and their services on the Internet.

Version Control

Features that maintain the integrity of a program or **Project** by controlling the ability of an individual to modify the program or Project, and providing an audit trail for accepted modifications.

WSDL

WSDL (Web Services Flow Language) is an **XML**-based language, derived from **SOAP**, used to describe the services a business offers via the Internet. WSDL provides the means of expressing business services in the **UDDI** registry.

XML

XML (Extensible Markup Language) is a superset of **HTML**, which also describes the content in terms of what data is being described. XML is *extensible* because—unlike HTML—the markup symbols are unlimited and self-defining.

XSD

XSD (XML Schema Definition) specifies how to formally describe the elements in an **XML** document. It is more powerful than, and generally replaces, the older Document Type Definition (DTD). eGate Integrator makes use of **Object Type Definition (OTD)**s described in XSD, as well as DTD.

XSLT

XSLT (Extensible Stylesheet Language Transformation) is a language for transforming XML documents into other XML documents. It is designed for use as part of XSL, which is a stylesheet language for XML. eGate Integrator makes use of **Collaboration Definition**s coded in XSLT.

eGate Integrator 5.0 Terminology

Table 121 lists terminology that is new with eGate Integrator release 5.0 along with equivalent terms from e*Gate release 4.x, where applicable.

Table 121 Terminology Cross-Reference

eGate 5.0 Term	Equivalent e*Gate 4.x Term
Connection	e*Way Connection
Connectivity Map	Schema Network View (closest)
Deployment	Running the Control Broker
Deployment Profile	<none> (part of Schema)
Enterprise Designer	Enterprise Manager
Enterprise Manager	Enterprise Monitor
Environment	Schema (physical layer only)
eWay	e*Way, e*Way Connection
eWay Configuration	e*Way Connection Configuration
External Application	e*Way Connection
External System	e*Way Connection
JMS Connection	e*Way Connection
ICAN Monitor	Enterprise Monitor
Integration Server	<none>
Link	JMS e*Way Connection
Linked Message Destination	<none>
Logical Host	Participating Host
Message Destination	Topic or queue
Message Server	MS IQ Manager
Object Type Definition (OTD)	Event Type Definition (ETD)
Process Manager	Control Broker
Project	Schema (logical layer only)
Queue	MS queue
Repository	Registry
Subproject	Schema (logical layer only)
Topic	JMS topic

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X