SUN SEEBEYOND

eWAY™ ADAPTER FOR SUN JAVA SYSTEM APPLICATION SERVER USER'S GUIDE

Release 5.1.3

Draft Version: Sun Microsystems Internal Use Only

Sun microsystems

Part Number: 819-7409-11

Version 20070411094249

Contents

Chapter 1

Introducing the Sun Java System Application Server eWay

This document describes how to install, configure, and implement the Sun SeeBeyond eWay™ Adapter for Sun Java System Application Server, in a typical Sun Java Composite Application Platform Suite environment.

This chapter provides a brief overview of operations and components, general features, and system requirements of the Sun SeeBeyond eWay™ Adapter for Sun Java System Application Server (also referred to as the Sun AppServer eWay through this document).

What's in This Chapter

1.1 The Sun Java™ System Application Server

The Sun Java System Application Server is Sun's J2EE 1.4 compatible platform for developing and delivering Java web services. The application server comes in three editions: the free **Platform** edition which is embedded in many third party systems and applications, **Standard** edition which adds operations management and monitoring, and **Enterprise** edition which also adds Sun's Always On technology, and provides enhanced load balancing and cluster management capabilities.

1.2 The Sun SeeBeyond eWay Adapter for Sun Java™ System Application Server

The Sun SeeBeyond eWay Adapter for Sun Java™ System Application Server (referred to as the Sun AppServer eWay throughout this document) is an application specific eWay that facilitates integration between applications built on a Sun Java™ System Application Server and eGate (Sun Java™ Composite Application Platform Suite) using the Enterprise Java Bean (EJB) component model.

1.3 What's New in This Release

The Sun SeeBeyond eWay™ Adapter for Sun Java System Application Server includes the following changes and new features:

New for version 5.1.3

This is a maintenance release. No new features.

New for version 5.1.2

- This is a maintenance release. No new features.

New for version 5.1.1

- This is a maintenance release. No new features.

New for Version 5.1.0

- Version Control: An enhanced version control system allows you to effectively manage changes to the eWay components.
- Multiple Drag-and-Drop Component Mapping from the Deployment Editor: The Deployment Editor now allows you to select multiple components from the Editor's component pane, and drop them into your Environment component.
- Support for Runtime LDAP Configuration: eWay configuration properties now support LDAP key values.
- MDB Pool Size Support: Provides greater flow control (throttling) by specifying the maximum and minimum MDB pool size.
- Connectivity Map Generator: Generates and links your Project's Connectivity Map components using a Collaboration or Business Process.
- Added support for Sun Java System Application Server version 8.1. Uses IIOP to reduce dependency on specific Application Server client libraries.

Many of these features are documented further in the *Sun SeeBeyond eGate™ Integrator User's Guide* or the *Sun SeeBeyond eGate™ Integrator System Administrator Guide*.

1.4 What's in This Document

The Sun SeeBeyond eWay™ Adapter for Sun Java System Application Server User's Guide includes the following chapters:

- **Chapter 1 "Introducing the Sun Java System Application Server eWay"** provides an overview of the Sun Java System Application Server and the SeeBeyond Sun Java System eWay Intelligent Adapter.

- **Chapter 2** **"Installing the Sun Java System Application Server eWay"** provides the supported operating systems and system requirements for the Sun AppServer eWay. It also includes directions for installing the Sun AppServer eWay and accessing the accompanying documentation and sample Projects.
- **Chapter 3** **"Sun Application Server Components"** describes various Sun Microsystem Java 2 Enterprise Edition (J2EE) applications and Sun Application Server technologies employed in the Sun Java System Application Server.
- **Chapter 4** **"Sun AppServer eWay Component Communication"** describes how the components of the Sun AppServer eWay communicate with the Sun Java System Application Server.
- **Chapter 5** **"Configuring the Sun Java™ System Application Server eWay Properties"** describes how to configure the Sun Java System Application Server eWay properties, and provides a list of the eWay properties and their required values.
- **Chapter 6** **"Using the Sun AppServer OTD Wizard"** describes how to use the Sun Java System AppServer OTD Wizard to create Object Type Definitions (OTDs).
- **Chapter 7** **"Implementing a Project Using eInsight"** describes how to create and implement the eInsight (Business Process) sample Project.
- **Chapter 8** **"Implementing a Project Using Java Collaboration Definitions (JCD)"** describes how to create and implement the Java Collaboration sample Project included with the eWay installation.

Classes and Methods

The Sun Java System AppServer eWay does not include a Javadoc. For information on classes and methods, refer to the Sun J2EE Javadoc at the following Web site:

http://java.sun.com/products/servlet/download.html

1.4.1 Scope of the Document

This user's guide provides a description of the Sun SeeBeyond eWay™ Adapter for Sun Java System Application Server. It includes directions for installing the eWay, configuring the eWay properties, and implementing the eWay's sample Projects. This document is also intended as a reference guide, listing available properties and functions.

1.4.2 Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning Java Composite Application Platform Suite system. This person must also understand any operating systems on which the Java Composite Application Platform Suite will be installed (Windows and UNIX), and must be thoroughly familiar with Windows-style GUI operations.

1.4.3 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

Text Convention	Used For	Examples
Bold	Names of buttons, files, icons, parameters, variables, methods, menus, and objects	▪ Click **OK**. ▪ On the **File** menu, click **Exit**. ▪ Select the **eGate.sar** file.
`Monospaced`	Command line arguments, code samples; variables are shown in ***bold italic***	`java -jar` ***`filename`***`.jar`
Blue bold	Hypertext links within document	See **Text Conventions** on page 10
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.5 Sun Microsystems, Inc. Web Site

The Sun Microsystems web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.sun.com

1.6 Documentation Feedback

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

CAPS_docsfeedback@sun.com

Chapter 2

Installing the Sun Java System Application Server eWay

This Chapter describes how to install the Sun Java System Application Server eWay Adapter, as well as the accompanying documentation and sample Projects.

What's in This Chapter

2.1 Sun Java Application Server eWay Requirements

The Sun Java Application Server eWay Readme contains the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements
- The Sun Java Application Server eWay Readme is uploaded with the eWay's documentation file (SunJavaSystemeWayDocs.sar) and can be accessed from the Documentation tab of the Sun Java Composite Application Platform Suite Installer. Refer to the Sun Java Application Server eWay Readme for the latest requirements before installing the Sun Java Application Server eWay.

2.2 Installing the Sun Java System Application Server eWay

The Sun Java Composite Application Platform Suite Installer, a web-based application, is used to select and upload eWays and add-on files during the installation process. The following section describes how to install the components required for this eWay.

When the Repository is running on a UNIX operating system, the eWays are loaded from the Sun Java Composite Application Platform Suite Installer running on a Windows platform connected to the Repository server using Internet Explorer.

2.2.1 Installing the eWay on a JavaCAPS Supported System

Follow the directions for installing the Sun Java Composite Application Platform Suite in the *Sun Java Composite Application Platform Suite Installation Guide. After you have installed eGate or eInsight, do the following:*

1. From the Sun Java Composite Application Platform Suite Installer's **Select Sun Java Composite Application Platform Suite Products to Install** table (Administration tab), expand the **eWay** option.
2. Select the products for your Sun Java Composite Application Platform Suite and include the following:
   - **FileeWay** (the File eWay is used by most sample Projects)
   - **SunJavaSystemeWay**

   To upload the Sun Java System Application Server eWay User's Guide, Help file, Readme, and sample Projects, select the following:
   - **SunJavaSystemeWayDocs**
3. Once you have selected all of your products, click **Next** in the top-right or bottom-right corner of the **Select Sun Java Composite Application Platform Suite Products to Install** box.
4. From the **Selecting Files to Install** box, locate and select your first product's SAR file. Once you have selected the SAR file, click **Next**. Your next selected product appears. Follow this procedure for each of your selected products. The **Installation Status** window appears and installation begins after the last SAR file has been selected.
5. Once your product's installation is finished, continue installing the Sun Java Composite Application Platform Suite as instructed in the *Sun Java Composite Application Platform Suite Installation Guide.*

Adding the eWay to an Existing Suite Installation

If you are adding the eWay to an existing Sun Java Composite Application Platform Suite installation, do the following:

1. Complete steps 1 through 4 above.
2. Once your product's installation is finished, open the Enterprise Designer and select **Update Center** from the Tools menu. The **Update Center Wizard** appears.
3. For Step 1 of the wizard, simply click **Next**.
4. For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field, then click **Next**.
5. For Step 3 of the wizard, wait for the modules to download, then click **Next**.
6. The wizard's Step 4 window displays the installed modules. Review the installed modules and click **Finish.**
7. When prompted, restart the IDE (Integrated Development Environment) to complete the installation.

After Installation

Once you install the eWay, it must then be incorporated into a Project before it can perform its intended functions. See the *Sun SeeBeyond eGate™ Integrator User's Guide* for more information on incorporating the eWay into an eGate Project.

2.2.2 Installing eWay Enterprise Manager plug-ins

The **Sun SeeBeyond Enterprise Manager** is a Web-based interface that allows you to monitor and manage your Java Composite Application Platform Suite applications. The Enterprise Manager requires an eWay specific "plug-in" for each of your installed eWays. These plug-ins enable the Enterprise Manager to target specific alert codes for each eWay type, as well as to start and stop the inbound eWays.

The *Sun Java Composite Application Platform Suite Installation Guide* describes how to install the Enterprise Manager. The *Sun SeeBeyond eGate™ Integrator System Administration Guide* describes how to monitor servers, services, logs, and alerts using the Enterprise Manager and the command-line client.

The **eWay Enterprise Manager plug-ins** are available from the **List of Components to Download** under the Sun Java Composite Application Platform Suite Installer's **DOWNLOADS** tab.

There are two ways to add the eWay Enterprise Manager plug-ins:

1 From the Enterprise Manager:

  A From the **Enterprise Manager**'s Explorer toolbar, click the **Configuration** icon.

  B Click the **Web Applications Manager** tab, go to the **Auto-Install from Repository** tab, and connect to your Repository.

  C Select the application plug-ins you require, and click **Install**. The application plug-ins are installed and deployed.

2 From the **Sun Java Composite Application Platform Suite Installer:**

  A From the **Sun Java Composite Application Platform Suite Installer's Download tab**, select the Plug-Ins you require and save them to a temporary directory.

  B Log onto the **Sun SeeBeyond Enterprise Manager**. From the **Enterprise Manager**'s Explorer toolbar, click the **Configuration** icon.

  C Click the **Web Applications Manager** tab and go to the **Manage Applications** tab.

  D Browse for and select the WAR file for the application plug-in that you downloaded, and click **Deploy**. The plug-in is installed and deployed.

Sun Java System Application Server eWay Alert Codes

You can view and delete alerts using the Enterprise Manager. An alert is triggered when a specified condition occurs in a Project component. The purpose of the alert is to warn the administrator or user that a condition has occurred.

To View the eWay Alert Codes

1. Add the eWay Enterprise Manager plug-in for this eWay.
2. From the Enterprise Manager's **Explorer** toolbar, click the **Configuration** icon.
3. Click the **Web Applications Manager** tab and go to the **Manage Alert Codes** tab. Your installed alert codes are displayed under the **Results** section. If your eWay alert codes are not available displayed under **Results**, do the following

   A. From the **Install New Alert Codes** section, browse to and select the eWay alert properties file for the application plug-in that you added. The alert properties files are located in the **alertcodes** folder of your Sun Java Composite Application Platform Suite installation directory.

   B. Click **Deploy**. The available alert codes for your application are displayed under **Results**. A listing of available this eWay's alert codes is displayed in Table 2.

Table 2 Sun Java System Application Server eWay Alert Codes

Alert Code	Description	User Action
EJB-ERRORENCOUNTERED000001	Error encountered in EJB eWay. Check debug logs for details.	An error has occurred, such as ▪ EJB has not been deployed. ▪ Configuration properties are not correct. ▪ JNDI name is invalid. ▪ OTD uses an incompatible version of the EJB. Refer to the log for more information.

For information on Managing and Monitoring alert codes and logs, see the *Sun SeeBeyond eGate Integrator System Administration Guide.*

2.3 ICAN 5.0 Project Migration Procedures

This section describes how to transfer your current ICAN 5.0 Projects to Sun Java Composite Application Platform Suite, version 5.1.3. Only Projects developed on ICAN 5.0.2 and above can be migrated successfully to the Sun Java Composite Application Platform Suite. To migrate your ICAN 5.0 Projects, do the following:

Export the Project

1. Before you export your Projects, save your current ICAN 5.0 Projects to your Repository.
2. From the Project Explorer, right-click your Repository and select **Export** from the shortcut menu. The Export Manager appears.
3. From the **Select Projects from the list field** of the Export Manager, select one or more Projects that you want to export and move them to the **Selected Projects** field

by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Projects.

4 In the same manner, from the **Select Environments from the list** field, select the Environments that you want to export and move them to the Selected Environments field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Environments.

5 Browse to select a destination for your Project ZIP file and enter a name for your Project in the **ZIP file** field.

6 Click **Export** to create the Project ZIP file in the selected destination.

Install Sun Java Composite Application Platform Suite

7 Install Sun Java Composite Application Platform Suite, including all eWays, libraries, and other components used by your ICAN 5.0 Projects.

8 Start the Sun SeeBeyond Enterprise Designer.

Import the Project

9 From the Enterprise Designer's Project Explorer, right-click the Repository and select **Import Project** from the shortcut menu. The Import Manager appears.

10 Browse to and select your exported Project file.

11 Click **Import**. A warning message, **"Missing APIs from Target Repository**," may appear at this time. This occurs because various product APIs were installed on the ICAN 5.0 Repository, when the Project was created, that are not installed on the Sun Java Composite Application Platform Suite Repository. These APIs may or may not apply to your Projects. You can ignore this message if you have already installed all of the components that correspond to your Projects. Click **Continue** to resume the Project import.

12 Close the Import Manager after the Project is successfully imported.

Deploy the Project

13 You must create a new Deployment Profile for each of your imported Projects. When you export a Project, the Project's components are automatically *"checked in"* to Version Control to write-protect each component. These protected components appear in the Explorer tree with a red padlock in the bottom-left corner of each icon. Before you can deploy the imported Project, the Project's components must first be *"checked out"* of Version Control from both the Project Explorer and the Environment Explorer. To *"check out"* all of the Project's components, do the following:

A From the Project Explorer, right-click the Project and select **Version Control > Check Out** from the shortcut menu. The Version Control - Check Out dialog box appears.

B Select **Recurse Project** to specify all components, and click **Check Out**.

C Select the Environment Explorer tab, and from the Environment Explorer, right-click the Project's Environment and select **Version Control > Check Out** from the shortcut menu.

D Select **Recurse Environment** to specify all components, and click **Check Out**.

14 If your imported Projects include File eWays, they must be reconfigured in your Environment prior to deploying the Project. To reconfigure your File eWays, do the following:

   A The Environment File External System properties can now accommodate both inbound and outbound eWays. If your previous Environment includes both inbound and outbound File External Systems, delete one of these (for example, the outbound File External System).

   B From the Environment Explorer tree, right-click your remaining File External System, and select **Properties** from the shortcut menu. The Properties Editor appears.

   C The Directory property has been relocated from the Connectivity Map Properties to the Environment Properties. Set the inbound and outbound Directory values, and click **OK**.

15 Deploy your Projects.

Note: *Only Projects developed on ICAN 5.0.2 and above can be imported and migrated successfully into the Java Integration Suite.*

Chapter 3

Sun Application Server Components

This section provides an overview of the various Sun Microsystems Java 2 Enterprise Edition (J2EE) Applications and Sun Java System Application Server technologies employed in the Sun Application Server.

What's in This Chapter:

3.1 Java Naming and Directory Interface (JNDI)

The JNDI service is a set of APIs published by Sun that interface with a directory to locate a named objects. APIs allow Java programs to store and lookup objects using multiple naming services in a standard manner. The naming service may be LDAP, a file system, or an RMI registry. Each naming service has a corresponding provider implementation that can be used with JNDI. The ability for JNDI to "plug in" any implementation for any naming service (or span across naming services with a federated naming service) easily provides another level of programming abstraction. This level of abstraction allows Java code using JNDI to be portable against any naming service. For example, no code changes should be needed by the Java client code to run against an RMI registry or an LDAP server.

The Sun Application Server Naming Service

Any J2EE compliant application server, such as the Sun Java System Application Server, has a JNDI subsystem. The JNDI subsystem is used in an Application Server as a directory for such objects as resource managers and Enterprise JavaBeans (EJBs). Objects managed by the Sun AppServer container have default environments for getting the JNDI **InitialContext** loaded when they use the default **InitialContext()** constructor. For a Collaboration using a Sun AppServer EJB Object Type Definition (OTD) to find the home interface of an EJB, the JNDI properties must be configured and associated with the OTD. However, for other external clients, accessing the Sun AppServer naming service requires a Java client program that sets up the appropriate JNDI environment when creating the JNDI Initial Context.

There are essentially two environments that have to be configured, **Context.PROVIDER_URL** and **Context.INITIAL_CONTEXT_FACTORY**. For Sun AppServer, the Context.PROVIDER_URL environment is

```
iiop://<serverhost>:<port>/
```

where <serverhost> is the hostname on which the Sun Application Server instance is running and <port> is the port at which the Webserver instance is listening for connections. For example:

```
iiop://localhost:3700/
```

The initial context factory class for the Sun AppServer JNDI is **com.sum.appserv.naming.CNCtxFactory**. This class should be supplied to the **Context.INITIAL_CONTEXT_Factory** environment property when constructing the initial context. The overloaded **InitialContext(Map) constructor** must be used in this case.

Sample Code

The following code is an example of creating an initial context to Sun AppServer JNDI from a stand-alone client:

```
HashMap env = new HashMap();
env.put (Context.PROVIDER_URL, "iiop://localhost:3700/");
env.put (Context.INITIAL_CONTEXT_FACTORY,
"com.sum.appserv.naming.CNCtxFactory");
Context initContext = new InitialContext (env);
…
```

3.2 Enterprise JavaBeans (EJBs)

Enterprise JavaBeans are server-side Java component architecture. Developers use EJBs to design and develop customized, reusable business logic. EJBs are the units of work that an application server is responsible for and exposes to the external world. The Sun Application Server provides the architecture for writing business logic components, allowing Web servers to easily access data.

Session Beans

Session Beans are business process objects that perform actions. An action may be opening an account, transferring funds, or performing a calculation. Session Beans consist of the remote, home, and bean classes. A client gets a reference to the Session Bean's home interface in order to create the Session Bean remote object, which is essentially the bean's factory. The Session Bean is exposed to the client with the remote interface. The client uses the remote interface to invoke the bean's methods. The actual implementation of the Session Bean is done with the bean class.

Entity Beans

Entity Beans are data objects that represent the real-life objects on which Session Beans perform actions. Objects may include items such as accounts, employees, or inventory. An Entity Bean, like a Session Bean, consists of the remote, home, and bean classes. The client references the Entity Bean's home interface in order to create the Entity Bean remote object (essentially the bean's factory). The Entity Bean is exposed to the client with the remote interface, which the client uses to invoke the bean's methods. The implementation of the Entity Bean is done with the bean class.

Chapter 4

Sun AppServer eWay Component Communication

This chapter provides an overview of how the components of the Sun AppServer eWay communicate with the Sun Java System Application Server.

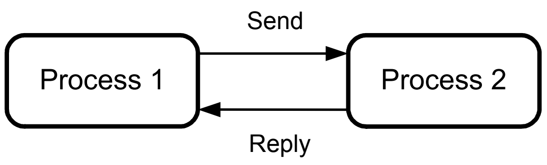
What's in This Chapter

4.1 Synchronous Communication

The Sun AppServer eWay takes advantage of Synchronous communication in message delivery. Synchronous messages provide outbound communication and require OTDs to hold the data structure and define rules referenced in the EJB.

Synchronous communication is considered an unbuffered process, requiring complete data transmission and reply or confirmation of message transmission failure before continuing with the process.This can be comparable to a phone call in which the caller makes the call and waits for a response before attempting to make another call. An example of synchronous communication is displayed in Figure 1.

Figure 1 Synchronous Communication



Synchronous Communication in eGate

- **eGate to Sun AppServer Transactions** – an outbound transaction, where ICAN makes a request to Sun Application Server and waits for a response. For more information, see “Synchronous Communication in eGate” on page 21.

4.2 Synchronous Communication in eGate

Synchronous communication carried out by the Sun AppServer eWay requires the creation of an OTD using the Sun Java System AppServer OTD Wizard. Sun AppServer OTDs are created using Sun AppServer's Session and Entity Beans EJB interface classes, that represent the methods of the EJB.

Once created, these methods are called from within a Collaboration, making them accessible to the user. The OTD queries the JNDI directory services and locates a home interface, uses the home interface to acquire remote interfaces, applies Iterator methods for managing multiple remote interface instances, and provides access to the remote interface methods. Collaborations can then be built between the OTD and OTDs for other applications, making the EJB methods available to that application.

4.2.1 The Sun Java System AppServer OTD

The Sun Java System AppServer OTD contains EJB methods that are callable from inside a Collaboration.

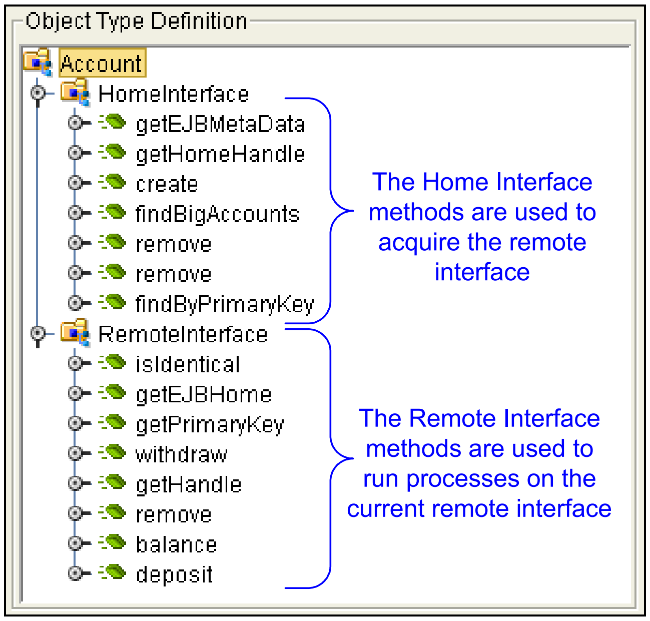
The OTD is divided into two portions:

- **Home Interface Methods** – used to acquire the Remote Interface, allowing OTDs to find and invoke EJB instances.

  For example, the home interface method **findBigAccounts()**, seen in Figure 2, could use the argument "balanceGreaterThan (100,000)" to find all account EJBs with an account balance over 100,000 and assign their remote interface to the Remote Instances OTD node.

- **Remote Interface Methods** – contains remote interface methods that allow processes to be run on the current remote interface.

Figure 2 EJB OTD nodes represent both Home and Remote Interface methods



Chapter 5

Configuring the Sun Java™ System Application Server eWay Properties

This chapter describes how to configure the Sun Java System Application Server eWay properties, and provides a list of the eWay properties and their required values.

What's in This Chapter

5.1 Configuring the Sun Java System Application Server eWay Properties

The Sun AppServer eWay includes a unique set of configuration parameters. After creating the eWays and the Sun AppServer External System in the Project's Environment, the property parameters can be modified for your specific system.

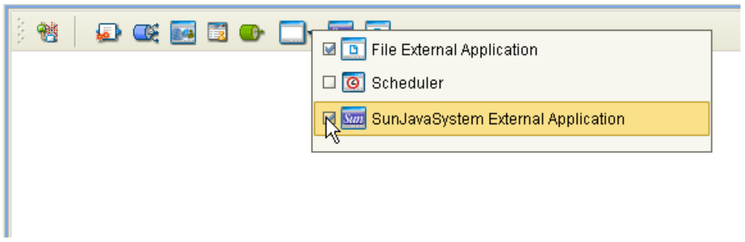
5.1.1 Selecting Sun Application Server as the External Application

To create a Sun AppServer eWay you must first create a Sun Java System AppServer External Application in your Connectivity Map. Sun AppServer eWays are located between a Sun AppServer External Application and a Service. Services are containers for Java Collaborations, Business Processes, eTL processes, and so forth.

To create the Sun Java System AppServer External Application

1. From the Connectivity Map toolbar, click the External Applications icon.
2. Select the **Sun Java System AppServer External Application** from the menu (see **Figure 3 on page 23**). The selected Sun AppServer External Application icon appears on the Connectivity Map toolbar.

Figure 3 External Applications Selection Menu

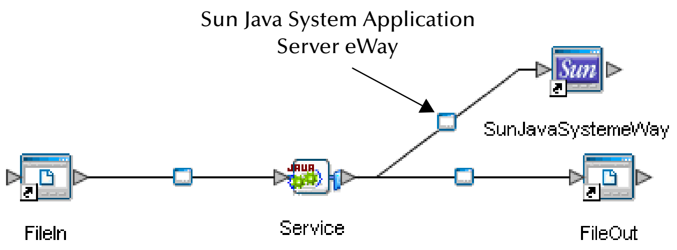


The new External System can now be dragged and dropped onto the Connectivity Map canvas and incorporated into a Project.

5.1.2 Accessing the eWay Properties

When you connect an External Application to a Collaboration, the Enterprise Designer automatically assigns the appropriate eWay to the link (Figure 4). Each eWay is supplied with a template containing default configuration properties that are accessible from the Connectivity Map and Environment Explorer Tree.

Figure 4 Connectivity Map with Components



5.1.3 Modifying the Sun Java System Application Server eWay Properties

The eWay properties can be modified after the eWays have been created in the Connectivity Map and the Project's Environment has been created. Sun AppServer eWay properties are modified from two locations: from the Connectivity Map and from the Environment Explorer tree.

Modifying the eWay Connectivity Map Properties

The Connectivity Map parameters most commonly apply to a specific component eWay, and may vary from other eWays (of the same type) in the Project.

1. From the Connectivity Map, double click the eWay icon, located in the link between the associated External Application and the Service.
2. The eWay **Properties Editor** opens with the Sun AppServer eWay Connectivity Map properties. Make any necessary modifications and click **OK** to save the settings.

Modifying the eWay Environment Properties

These parameters are commonly global, applying to all eWays (of the same type) in the Project. The saved properties are shared by all eWays for the specified External System.

1. From the Environment Explorer tree, right-click the Sun Java System AppServer External System. Select **Properties** from the shortcut menu. The **Properties Editor** opens with the Sun AppServer eWay Environment properties.
2. Make any necessary modifications to the Environment properties, and click **OK** to save the settings.

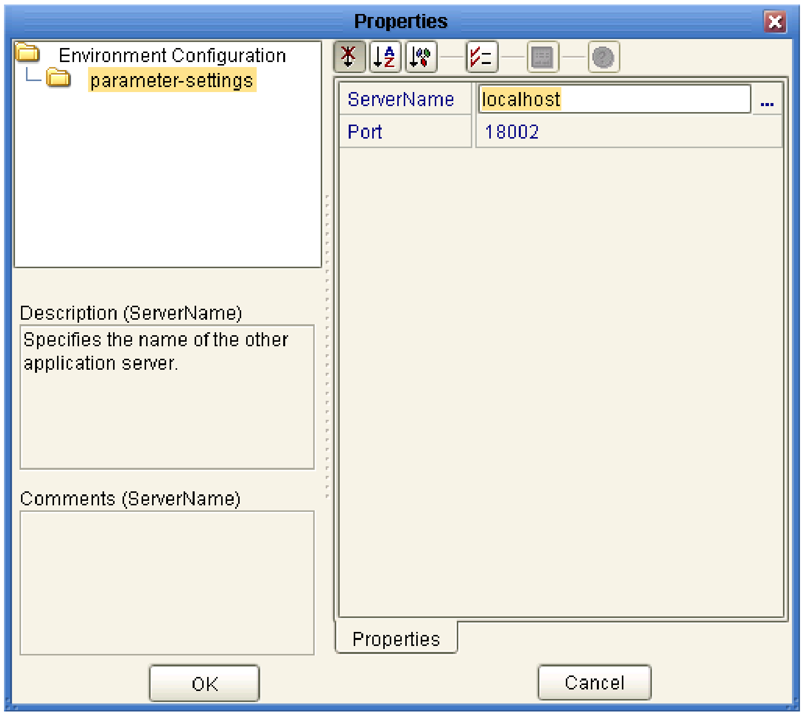
5.1.4 Using the Properties Editor

Modifications to the eWay properties are made using the Sun Java System AppServer eWay Properties Editor.

Modifying the Default eWay Properties

1. From the upper-left pane of the Properties Editor, select a subdirectory of the Properties tree. The parameters contained in that subdirectory are now displayed in the right pane of the Properties Editor. For example, clicking on the **parameter-settings** subdirectory displays the editable parameters in the right pane, as shown in Figure 5

Figure 5 Properties Editor: Sun AppServer eWay



ServerName	localhost
Port	18002

Description (ServerName)
Specifies the name of the other application server.

Comments (ServerName)

2. Click on any property field to make it editable. For example, click on the **ServerName** property to edit the properties settings. If a parameter's value is true/false or multiple choice, the field reveals a submenu of property options.

Click on the ellipsis (. . .) in the properties field (displayed when you click on the field). A separate configuration dialog box appears. This is helpful for large values that cannot be fully displayed in the parameter's property field. Enter the property value in the dialog box and click **OK**. The value now appears in the property field.

3 A description of each parameter is displayed in the **Description** pane when that parameter is selected, providing an explanation of any required settings or options.

4 The **Comments** pane provides an area for recording notes and information about the currently selected parameter. This is saved automatically for future referral.

5 Click **OK** to close the Properties Editor and save the changes.

5.2 Specifying JNDI Names

The Connectivity Map and Environment properties are used to specify JNDI names. There are two methods you can use to specify a JNDI name:

- Specify the **server name** and **port number** in the Environment properties, and specify a **JNDI name** in the Connectivity Map, **ejbJndiName** property. The eWay concatenates the server name and port number with the JNDI name and provides the qualified JNDI name.

  For example, if the specified server name is **MyServer**, the specified port number is **1111**, and the JNDI name is **ejb/MyStorageBin**, the eWay uses these property to construct the qualified JNDI name:

  ```
  corbaname:iiop:1.2@MyServer:1111#ejb/MyStorageBin
  ```

- Specify a qualified JNDI name in the Connectivity Map property, **ejbJndiName**. This name supersedes any values specified in the Environment properties.

5.3 Sun Java Systems Application Server eWay Properties

The Sun AppServer eWay properties are organized into the following sections:

- **Sun AppServer eWay Connectivity Map Properties** on page 26
- **Sun AppServer eWay Environment Properties** on page 27

5.4 Sun AppServer eWay Connectivity Map Properties

The eWay properties, accessed from the Connectivity Map, are organized into the following sections:

parameter-settings (Connectivity Map) on page 26

5.4.1 parameter-settings (Connectivity Map)

The **parameter-settings** section of the Connectivity Map properties contains the top level parameters displayed in Table 3.

Table 3 Connectivity Map - parameter-settings

Name	Description	Required Value
EjbJndiName	Specifies the JNDI name of the EJB on the remote server including any JNDI prefixes for the EJB interoperability. If the EJB and your Project EAR file are deployed on the same Sun Java System Application Server or the Sun Seebeyond Integration Server, you can use the **localEJB:** prefix as the qualified JNDI name. For example: **localEJB:ejb/MyStorageBin** without the corbaname/IIOP syntax. If a qualified JNDI name is specified in this property, this supersedes any values specified in the Environment properties: ServerName and Port (see **"Specifying JNDI Names" on page 25**.	Enter a JNDI name (String) or a qualified JNDI name.

5.5 Sun AppServer eWay Environment Properties

The eWay properties, accessed from the Environment Explorer tree, are organized into the following sections:

5.5.1 parameter-settings (Environment)

The **parameter-settings** section of the Environment properties contains the top level parameters displayed in Table 4.

Table 4 Environment - **parameter-settings**

Name	Description	Required Value
ServerName	Specifies the name of the other application server. If a qualified JNDI name is specified in the Connectivity Map property, EjbJndiName, that value supersedes any values specified in the Environment properties: ServerName and Port (see **“Specifying JNDI Names” on page 25**.	A name of an application server. The configured default is **localhost**.
Port	Specifies the Corba Port of the other application server. See **“Specifying JNDI Names” on page 25**.	An integer indicating a port number. The configured default is **18002**.

Chapter 6

Using the Sun AppServer OTD Wizard

Object Type Definitions define external data formats that characterize the input and output data structures in a Collaboration Definition. This chapter describes how to build and use Object Type Definitions (OTDs) using the Sun Java System AppServer OTD Wizard. JNI methods and inner classes are not supported.

What's in This Chapter

Note: *Consult the Sun Microsystems Javadoc for the latest API documentation regarding certain restrictions imposed by the J2EE specification.*

6.1 Creating A Sun Java System Application Server OTD

One important component of the Sun Java System Application Server eWay is the **SunJavaSystem AppServer OTD Wizard** which allows you to create Object Type Definition from EJB class files.

Note: *Java classes provided in the SunJavaSystem AppServer OTD Wizard can contain APIs created using the standard Sun JDK 1.3.x or JDK 1.4.x, but they must be compatible with either versions of the JVM. For example, Java code that is dependent on the JDK 1.3.x characteristic for java.util.TimeZone and java.util.SimpleTimeZone might not work with the same behavior or load correctly for JVM 1.4.x.*

Consult the Sun Microsystems Javadoc for the latest API documentation for specific restrictions imposed by the J2EE specification.

Caution: *JNI methods and inner classes are not supported.*

Important: *If the home or remote interface class, or their dependent classes contain recursive references, support for the corresponding EJB methods will be limited.*

Steps required to create an OTD include:

- **Select Wizard Type** on page 29

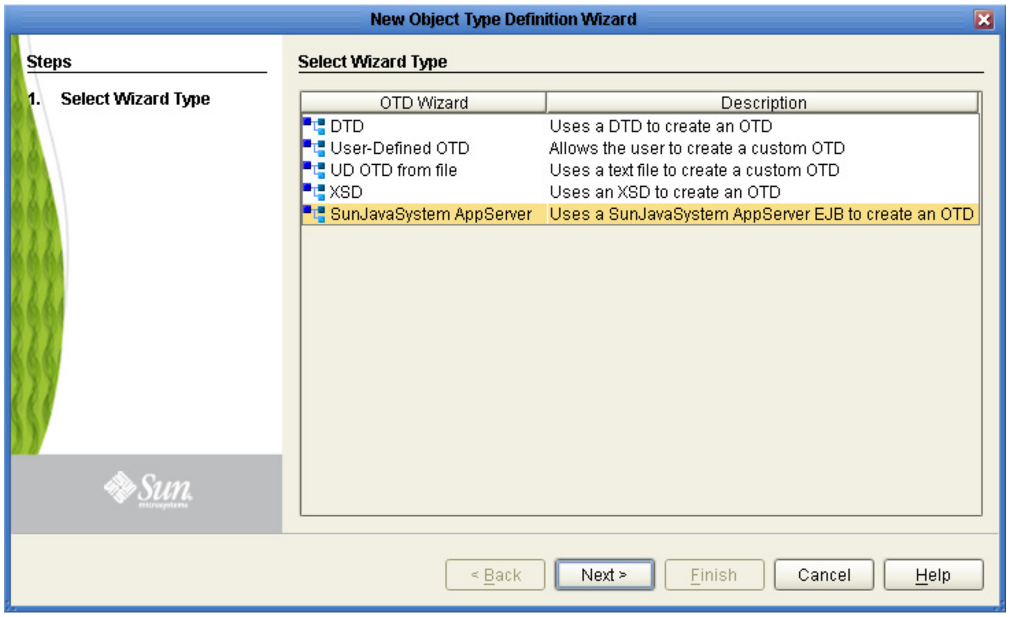
- **Specify OTD Name** on page 29
- **Select Code Base** on page 30
- **Select Home and Remote Interfaces** on page 31
- **Select Method Arguments** on page 32
- **Generate the OTD** on page 33

Select Wizard Type

1 From the Project Explorer tree, right click the Project and select **New > Object Type Definition** from the menu.

2 The **Select Wizard Type** page appears, displaying the available **OTD** wizards (see Figure 6).

Figure 6 OTD Wizard Selection



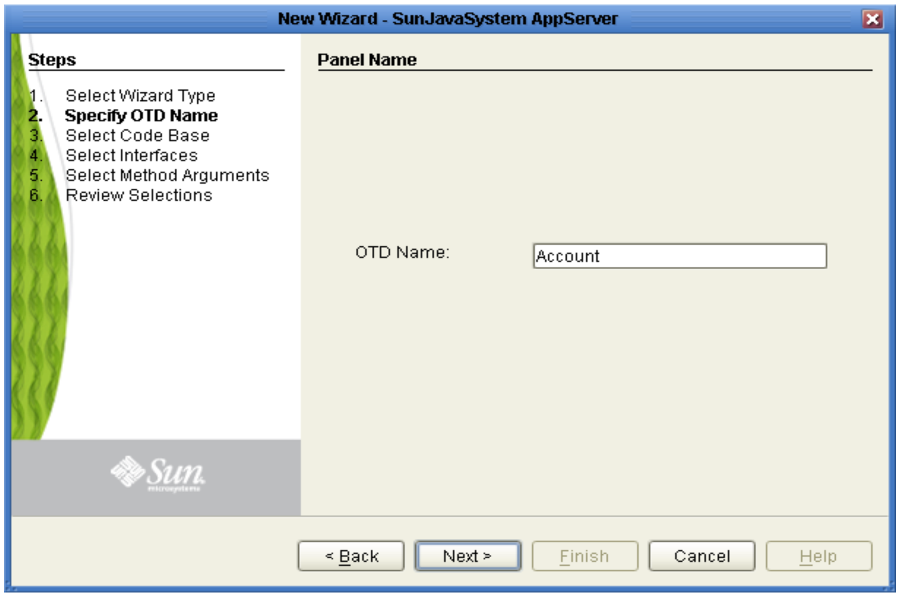
OTD Wizard	Description
DTD	Uses a DTD to create an OTD
User-Defined OTD	Allows the user to create a custom OTD
UD OTD from file	Uses a text file to create a custom OTD
XSD	Uses an XSD to create an OTD
SunJavaSystem AppServer	Uses a SunJavaSystem AppServer EJB to create an OTD

3 From the list, select the **Sun Java System AppServer** OTD Wizard and click **Next**. The **Specify OTD Name** page appears.

Specify OTD Name

4 Enter a name for the new OTD (see **Figure 7 on page 30**).

Figure 7 Specify OTD Name



OTD Name: Account

5 Click **Next**, the Select Home and Remote Interfaces page appears.

Select Code Base

From the **Select Code Base** page, you can select the directory that contains your EJB class files by selecting the root directory above the top-level Java package, or you can select a specific JAR file containing the EJB class files.

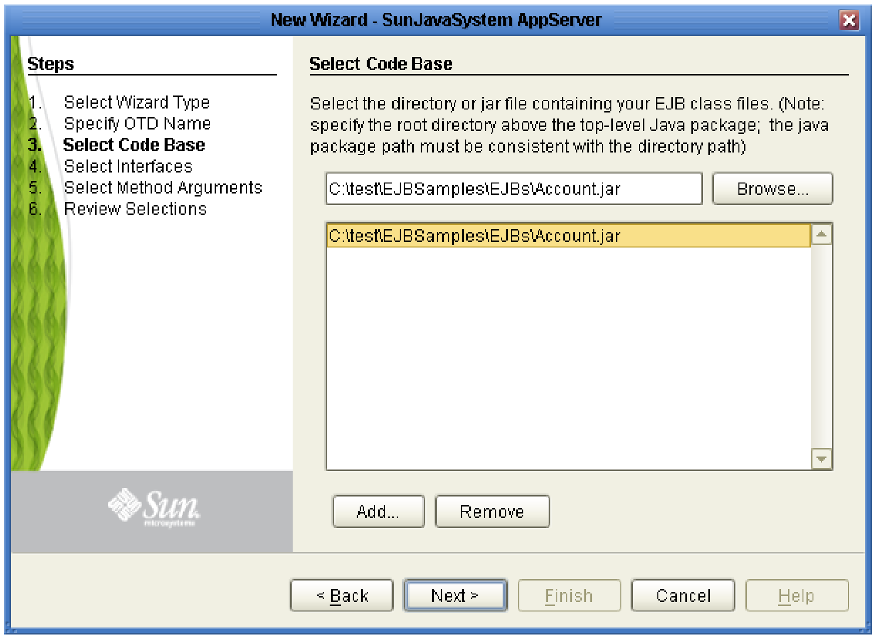
6 To select the root directory or archive file that contains the EJB class files, click **Browse** and navigate to and select the specific root directory or archive file. The EJB archive file must contain, at a minimum, both the Home and Remote Interface class pair for the EJB. Click **Add or Remove** to select or unselect any additional archive files to be available for design time (that is, in the JCD) and runtime (that is, in the Project EAR file). With the file or directory selected (at a minimum) click **Next** (see **Figure 8 on page 31**).

If you do not specify a JAR file, the program searches through the entire directory looking for all Java Archive files. Searches on top level drives or directories can significantly increase search times. Only recognized Class File Root file names are accepted in the File Name field.

EJB class files must contain a package name for the EJB Home and Remote Interface Classes. The wizard does not support EJB class files that do not provide a package name.

Note: *EJB Class files and their dependent files must be located in a directory to be used with the OTD Wizard. The Wizard does not support EJB classes imbedded in archived files, such as EAR, WAR, or SAR files. These files must be extracted to a directory before they can be used to create an OTD.*

Figure 8 Select Code Base page



Select the directory or jar file containing your EJB class files. (Note: specify the root directory above the top-level Java package; the java package path must be consistent with the directory path)

C:\test\EJBSamples\EJBs\Account.jar

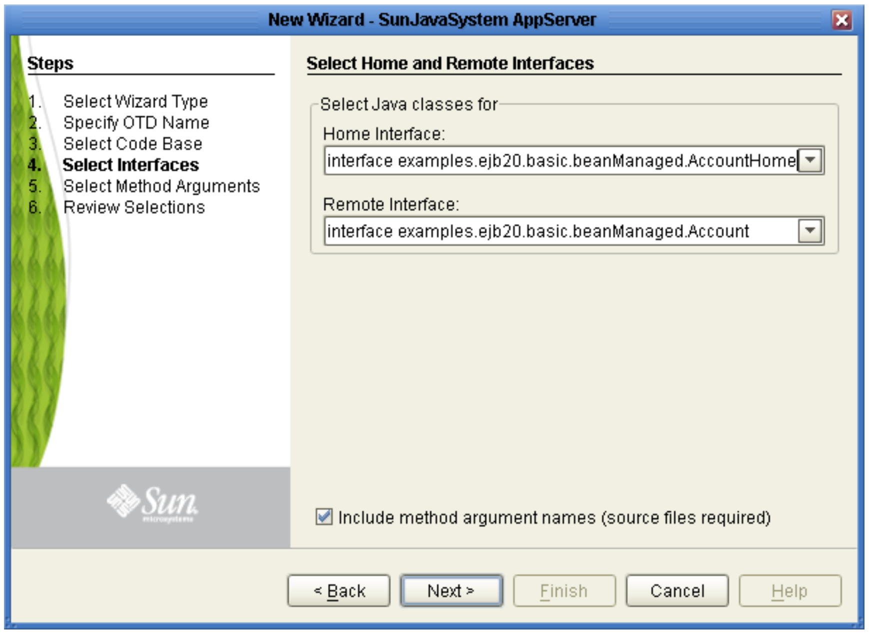
C:\test\EJBSamples\EJBs\Account.jar

Select Home and Remote Interfaces

The **Select Home and Remote Interfaces** page displays the selected Java Home and Remote Interfaces. These fields are automatically populated. Both fields include a drop-down list that allows you to select the appropriate home and remote interface (if more than one choice is available).

7 Review the selected Home and Remote Interface fields (see **Figure 9 on page 32**).

Figure 9 Select Home and Remote Interfaces page



Home Interface: interface examples.ejb20.basic.beanManaged.AccountHome

Remote Interface: interface examples.ejb20.basic.beanManaged.Account

☑ Include method argument names (source files required)

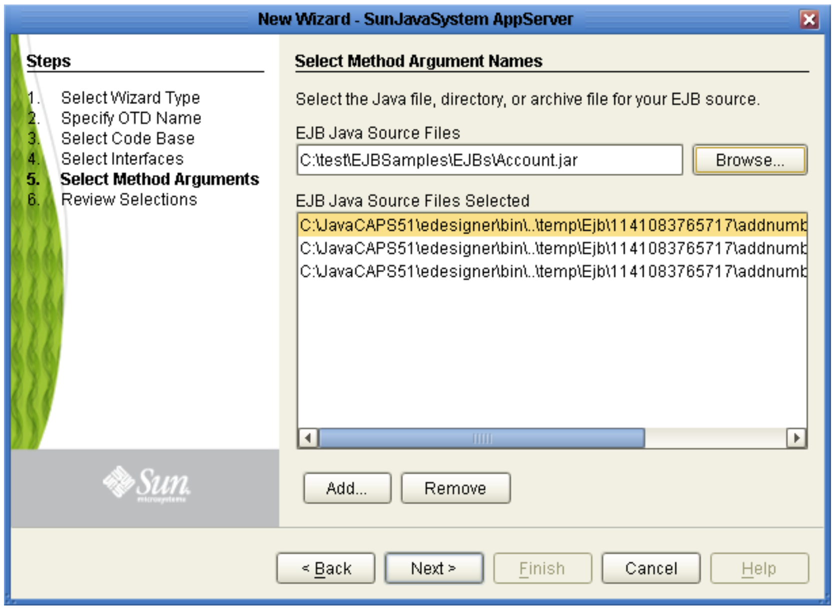
8 To add the method argument names to the OTD, select the Include method argument names option. Do not select this option if you do not have the EJB source code. Click **Next**. The Wizard advances to the **Select Methods Arguments** page.

If the selected EJB file's interfaces are valid and the Select Method argument names option was not selected, the wizard advances to the **Review Selections** page.

Select Method Arguments

9 The **Select Method Arguments Names** page appears. Enter the Java source file, or click **Browse** to locate the Java source files for the EJB archive supplied. Only a .java file or an archive file (containing .java files) is accepted; if a directory is supplied, then it searches only for .java files (see Figure 10).

Figure 10 Select Method Argument Names page


New Wizard - SunJavaSystem AppServer
Steps
1. Select Wizard Type
2. Specify OTD Name
3. Select Code Base
4. Select Interfaces
5. Select Method Arguments
6. Review Selections
Select Method Argument Names
Select the Java file, directory, or archive file for your EJB source.
EJB Java Source Files
C:\test\EJBSamples\EJBs\Account.jar
Browse...
EJB Java Source Files Selected
C:\JavaCAPS51\edesigner\bin\..\temp\Ejb\1141083765717\addnumb
C:\JavaCAPS51\edesigner\bin\..\temp\Ejb\1141083765717\addnumb
C:\JavaCAPS51\edesigner\bin\..\temp\Ejb\1141083765717\addnumb
Add...
Remove
< Back
Next >
Finish
Cancel
Help

10 Select the source files and click Open. The files are populated to the **EJB Java Source Files Selected** field. At least two files must be supplied -- one for the home interface and one for the remote interface, or the EJB bean implementation source. Once the field contains all the necessary EJB source files, click **Next**.

Caution: *It is the user's responsibility to match the correct source files to the EJB.*

Review Selections

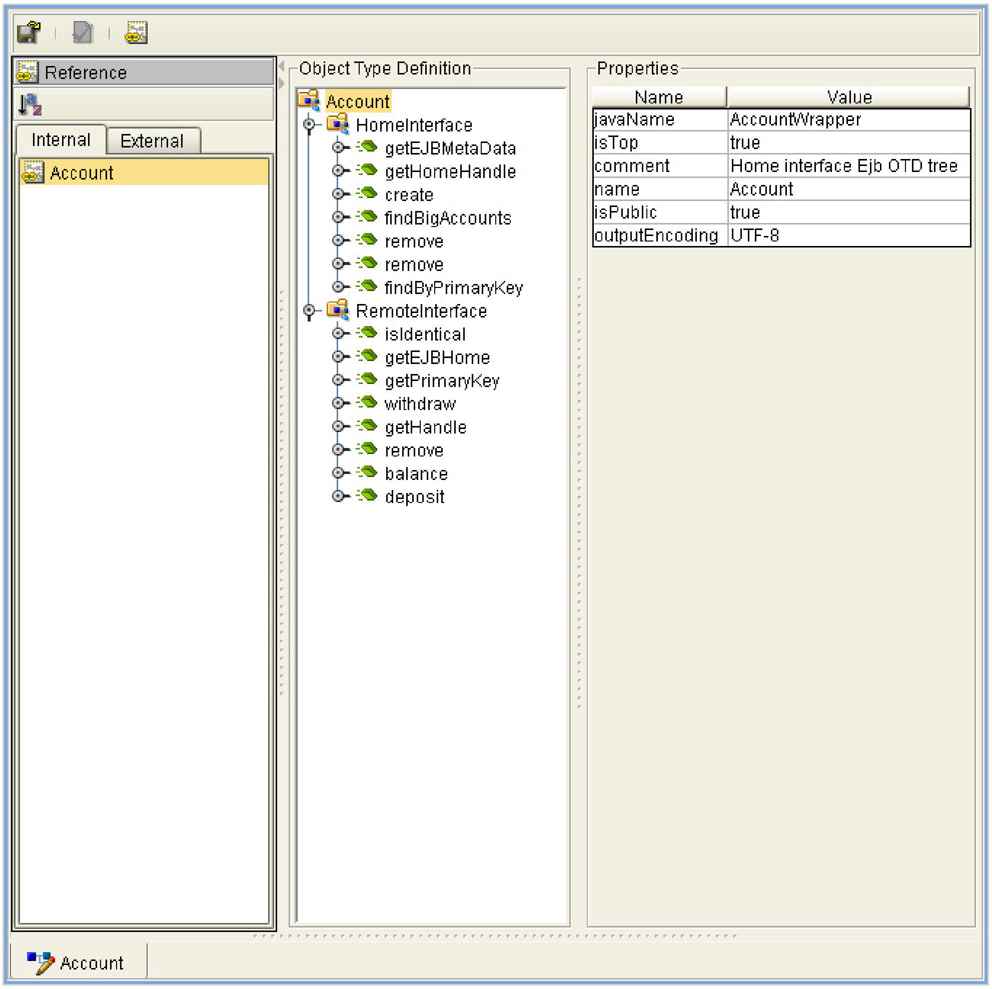
11 The Review Selections page appears. Review your selection information in the right pane of the page. To change any entries, click **Back** and return to the appropriate step.

Generate the OTD

12 Once you are satisfied with your selection information, click **Finish**. A **Confirm duplicate file** dialog box might appear at this point, stating that the JAR file is already in the Project. Select **Yes**, the JAR file is added to your Project. If you click **No**, an additional copy of the JAR file will be added to the Project.

The OTD Editor appears with the generated OTD (see Figure 11).

Figure 11 OTD Editor - New OTD



Name	Value
javaName	AccountWrapper
isTop	true
comment	Home interface Ejb OTD tree
name	Account
isPublic	true
outputEncoding	UTF-8

Chapter 7

Implementing a Project Using eInsight

This chapter describes how to use the Sun AppServer eWay with the Sun Java Composite Application Platform Suite's eInsight Business Process Manager and the Web Services interface.

Note: *You must have the* **eInsight.sar** *file installed to use the Web Services interface.*

What's in This Chapter

7.1 The eInsight Engine and Components

You can deploy an eGate component as an Activity in an eInsight Business Process. Once you have associated the desired component with an Activity, the eInsight engine can invoke it using a Web Services interface. Examples of eGate components that can interface with eInsight in this way are:

- Object Type Definitions (OTDs)
- eWays
- Collaborations

Using the eGate Enterprise Designer and eInsight, you can add an Activity to a Business Process, then associate that Activity—such as an eWay—with an eGate component. When eInsight runs the Business Process, it automatically invokes that component via its Web Services interface.

See the *Sun SeeBeyond eInsight™ Business Process Manager User's Guide* for details.

7.2 The Sun AppServer eWay With eInsight

An eInsight Business Process Activity can be associated with the Sun AppServer eWay during the system design phase. To make this association, select the desired operators under the eWay in the Enterprise Explorer and drag it onto the eInsight Business Process Designer canvas.

The operation is automatically changed to an Activity with an icon identifying the component that is the basis for the Activity. At run time, eInsight invokes each step in the order defined by the Business Process. Using eInsight's Web Services interface, the Activity in turn invokes the Sun Java System AppServer eWay.

7.3 The Sun AppServer eWay eInsight Sample Project

The following pages provide directions for creating the **prjSunAppServer_Sample_BPEL** Project. The prjSunAppServer_Sample_BPEL Project demonstrates how the Sun AppServer eWay uses Business Processes to provide the Business Logic.

Sample Overview

The Sun AppServer eWay Project, **prjSunAppServer_Sample_BPEL**, demonstrates the following:

- The inbound File eWay subscribes to an external input directory. When a target message is present the File eWay picks up the message that contains an item number (777) and publishes the message to the bpCallEJB Business Process.
- The Business Process uses the Sun Java System Application Server eWay to query the Sun Application Server External System for the item's ID and quantity, which it takes from the PointBase database provided with the Sun Java System Application Server Sample Bundle. The Business Process then writes this information to a message and publishes it to the outbound File eWay.
- The outbound File eWay publishes the new message to an external output directory.

For more information on creating Sun Java Composite Application Platform Suite Projects see the *Sun SeeBeyond eInsight™ Business Process Manager User's Guide* and the *Sun SeeBeyond eGate™ Integrator User's Guide*.

7.4 Sun Java System Application Server eWay Concerns

If an EJB you are using makes use of any complex type Java Classes (those types not supported directly by BPEL), it may be necessary to create your business logic in a Collaboration using the Collaboration Editor (Java), and calling the Collaboration from your Business Process. In this case the eInsight Business Process must first invoke the Java Collaboration. The Java Collaboration can then execute the call. For more information on calling a Java Collaboration from an eInsight Business Process, see the *Sun SeeBeyond eInsight Business Process Manager User's Guide*.

7.5 Importing a Sample Project

Sample eWay Projects are included as part of the installation package. To import a sample eWay Project to the Enterprise Designer do the following:

1. The sample files are uploaded with the eWay's documentation SAR file and downloaded from the Sun Java Composite Application Platform Suite Installer's **Documentation** tab. The **Sun_AppServer_eWay_Sample.zip** file contains the various sample Project ZIP files. Extract the samples to a local file.
2. Save all unsaved work before importing a Project.
3. From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import** from the shortcut menu. The **Import Manager** appears.
4. Browse to the directory that contains the sample Project zip file. Select the sample file (for this sample, **SunAppServer_Sample_BPEL.zip**) and click **Import**. After the sample Project is successfully imported, click **Close**.
5. Before an imported sample Project can be run you must do the following:
   - Create an **Environment** (see **"Creating an Environment" on page 48**)
   - Configure the eWays for your specific system (see **"Configuring the eWays" on page 48**)
   - Create a **Deployment Profile** (see **"Creating the Deployment Profile" on page 51**)
   - Create and start a domain (see **"Creating and Starting the Domain" on page 52**)
   - Build and deploy the Project (see **"Building and Deploying the Project" on page 52**)

7.6 Creating the prjSunAppServer_Sample_BPEL Project

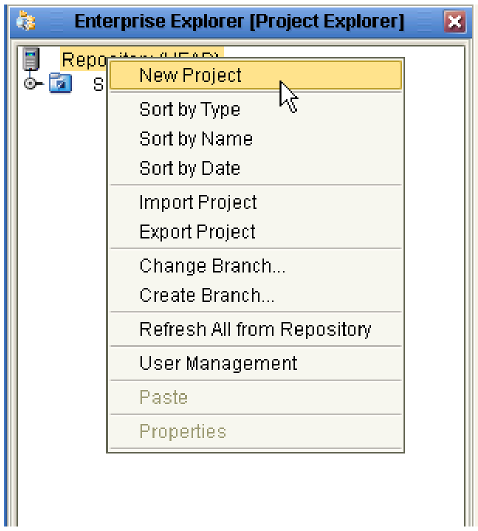
The following pages provide step by step directions for manually creating the **prjSunAppServer_Sample_BPEL** Project.

7.6.1 Creating a Project

The first step is to create a new Project in the SeeBeyond Enterprise Designer.

1. From the Enterprise Designer's Project Explorer tree, right-click the Repository and select **New Project** (see **Figure 12 on page 38**). A new Project (**Project1**) appears on the Project Explorer tree.

Figure 12 Enterprise Explorer - New Project



2. Rename the **Project1** Project to **prjSunAppServer_Sample_BPEL**.

7.6.2 Create an Object Type Definition Using the OTD Wizard

The Sun Java System AppServer OTD Wizard creates an Object Type Definition from a deployed EJB or JAR file.

Prepare the sample file bmp-simpleClient.jar

The JAR file used by this sample, **bmp-simpleClient.jar**, is part of the Sun Java System Application Server BMP sample. The BMP sample is located in the following location:

```
SunAppServerHome\samples\ejb\stateless\apps\simple\simple-ejb
```

Deploy the **bmp-simple.ear** file from the Sun Java System Application Server. Be sure to select the option to **generate RMIStubs**. After the EAR file has been deployed, the bmp-simpleClient.jar file is located in the following directory:

```
SunAppServerHome\domains\domain1\applications\j2ee-apps\bmp-simple
```

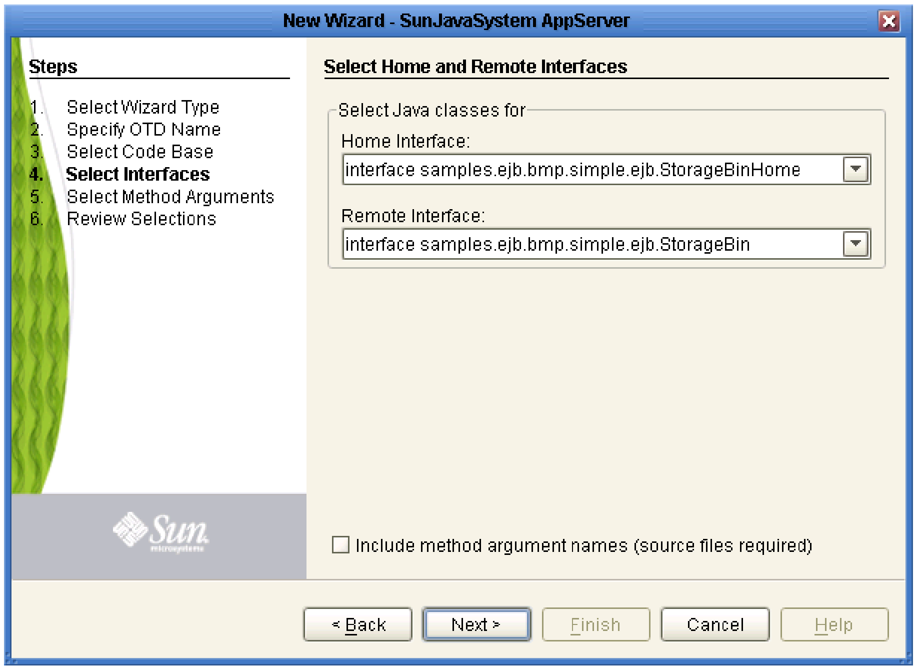
where *SunAppServerHome* is the location of your Sun Java System Application Server installation.

Create an OTD from bmp-simpleClient.jar

1. From the Project Explorer tree, right-click the **prjSunAppServer_Sample_BPEL** Project and select **New** > **Object Type Definition** from the shortcut menu. The **Object Type Definition Wizard** appears.
2. From the Select Wizard Type box, select **SunJavaSystem AppServer** Wizard and click **Next**.
3. For **step 2** of the wizard (**Specify OTD Name**), enter **Storage** as the OTD Name and click Next.
4. For **step 3** of the wizard (**Select Code Base**), browse to the location of the Sun Java System Application Server sample JAR file, **bmp-simpleClient.jar** (see **"Prepare the sample file bmp-simpleClient.jar" on page 38** for the JAR file location) and click **Open**. Now click **Add** and select the **bmp-simpleClient.jar file** and click **Open**. The file is added to the **Add/Remove** field. Click Next.
5. For step 4 of the wizard, **Select Interfaces** (see Figure 13), select the Home and Remote Interfaces as follows:
   - **Home Interface:** samples.ejb.bmp.simple.ejb.StorageBinHome
   - **Remote Interface:** interface samples.ejb.bmp.simple.ejb.StorageBin

   Click **Next**.

Figure 13 Sun Java System AppServer OTD Wizard - Step 4



Select Home and Remote Interfaces

Select Java classes for

Home Interface: interface samples.ejb.bmp.simple.ejb.StorageBinHome

Remote Interface: interface samples.ejb.bmp.simple.ejb.StorageBin

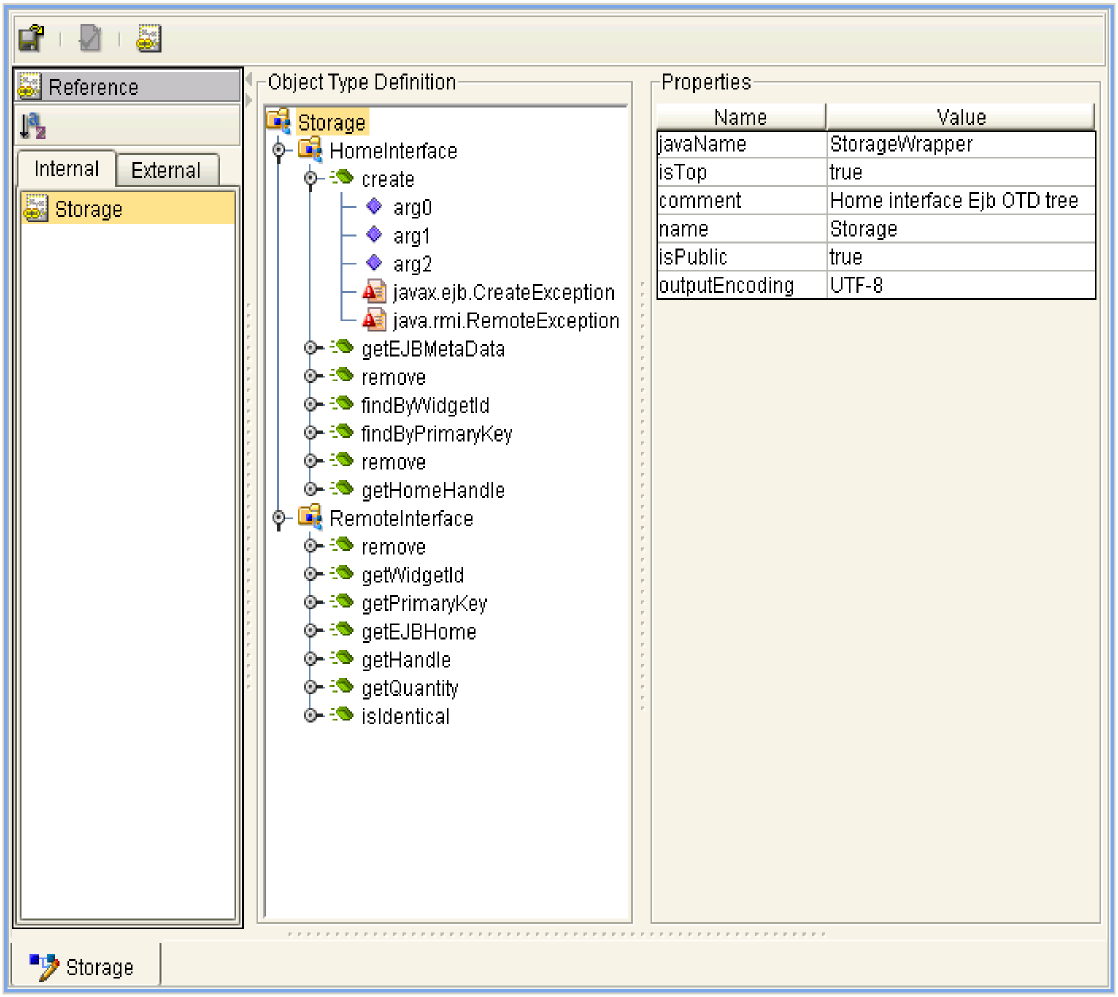
☐ Include method argument names (source files required)

6. The wizard proceeds to step 6, **Review Selections**. Review your selected information. When you are satisfied with your selections, click **Finish**.
7. Click **Next**. A **Confirm duplicate file** dialog box might appear at this point, stating that the bmp-simpleClient.jar is already in the Project. Select **Yes**, the bmp-

simpleClient.jar is added to your Project. If you click **No**, an additional copy of the JAR file will be added to the Project.

8 The OTD Editor appears with the New Storage OTD (See **Figure 14 on page 40**).

Figure 14 OTD Editor - Storage OTD



Name	Value
javaName	StorageWrapper
isTop	true
comment	Home interface Ejb OTD tree
name	Storage
isPublic	true
outputEncoding	UTF-8

For more information on the Sun Java System AppServer OTD Wizard see **"Using the Sun AppServer OTD Wizard" on page 28**

7.6.3 Creating the Business Process

The prjSunAppServer_Sample_BPEL contains one Business Process, **bpCallEJB**. To create this Business Process, do the following:

1 Right-click the **prjSunAppServer_Sample_BPEL** Project in the Project Explorer, and select **New** > **Business Process** from the shortcut menu. The eInsight Business Process Designer appears and **BusinessProcess1** is added to the Project Explorer tree. Rename **BusinessProcess1** to **bpCallEJB**.

2 From the Project Explorer tree, expand **SeeBeyond > eWays > File > FileClient**. Also, from the **prjSunAppServer_Sample_BPEL** Project, expand the **Storage** OTD node.

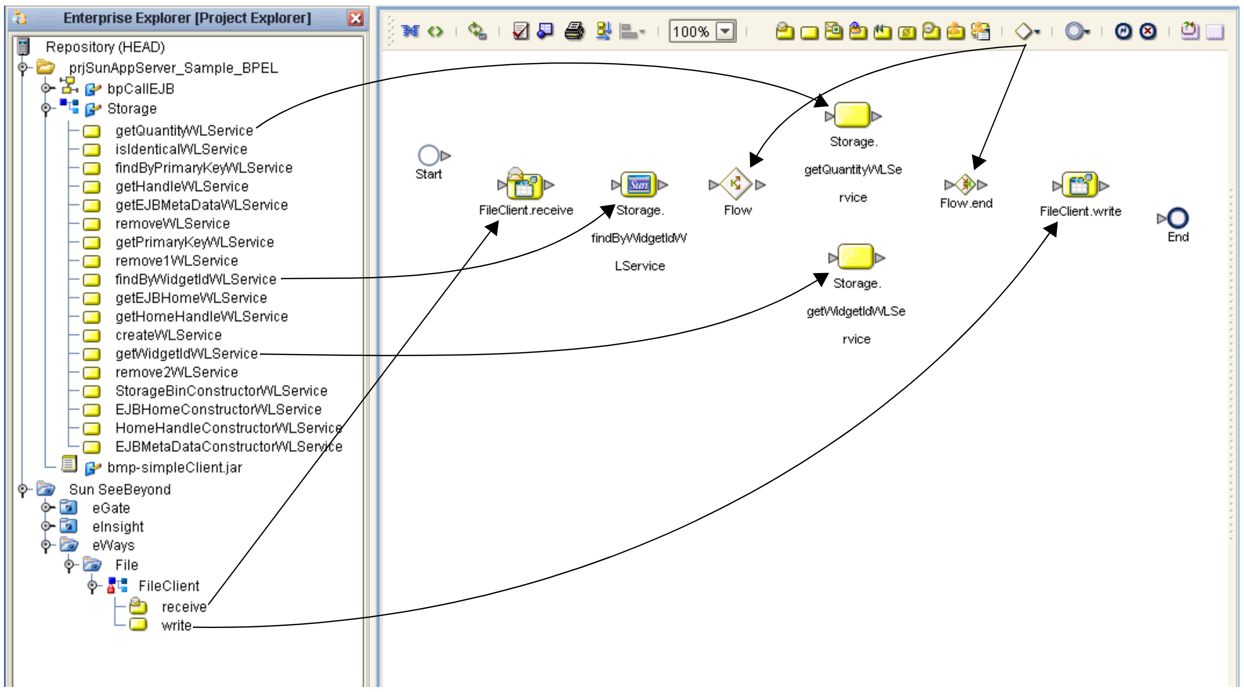
3 Populate the eInsight Business Process Designer canvas with the following activities from the Project Explorer tree, as displayed in Figure 15.

- **receive**, under SeeBeyond > eWays > File > FileClient
- **findByWidgetIdWLService**, under Storage
- **getQuantityWLService**, under Storage
- **getWidgetIdWLService**, under Storage
- **write**, under SeeBeyond > eWays > File > FileClient

Note: *It is always necessary to invoke an EJB home method (lookup, create, and so forth). In this case that home method is findByWidgetIdWLService.*

4 From the eInsight Business Process Designer toolbar, Branching Activities selection menu, drag **Flow** to the Business Process Designer canvas. The **Flow** and **Flow.end** Activities appear. The Flow activity allows you to specify one or more Business Process paths to be performed concurrently (see Figure 15).

Figure 15 eInsight Business Process Designer - Populate the Canvas

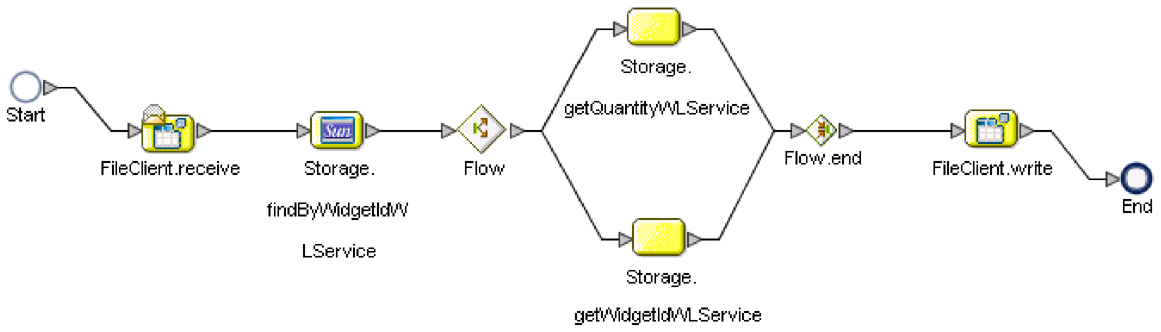


5 Link the elements by clicking on the element connector and dragging the cursor to the next element connector, making the following links as displayed in **Figure 16 on page 42**.

- Start -> FileClient.receive
- FileClient.receive -> Storage.findByWidgetIdWLService

- Storage.findByWidgetIdWLService -> Flow
- Flow -> Storage.getQuantityWLService
- Flow -> Storage.getWidgetIdWLService
- Storage.getQuantityWLService -> Flow.end
- Storage.getWidgetIdWLService -> Flow.end
- Flow.end -> FileClient.write
- FileClient.write -> End

Figure 16 Business Process Designer - Link the Modeling Elements



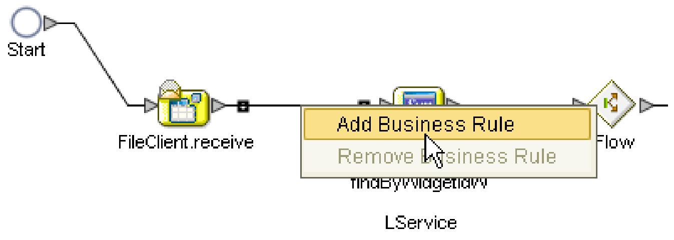
Configuring the bpCallEJB Modeling Elements

Business Rules, located between the Business Process Activities, allow you to configure the relationships between the input and output attributes of the Activities. Business Processes are created using the Business Process Designer's Business Rule Designer. To create the **bpCallEJB** Business Rules do the following:

Adding Business Rules

1 Right-click the link between the **FileClient.receive** and **Storage.findByWidgetIdWLService** Activities and select **Add Business Rule** from the shortcut menu (see Figure 17). A Business Rule Icon is added to the link.

Figure 17 eInsight Business Process Designer - Adding Business Rules



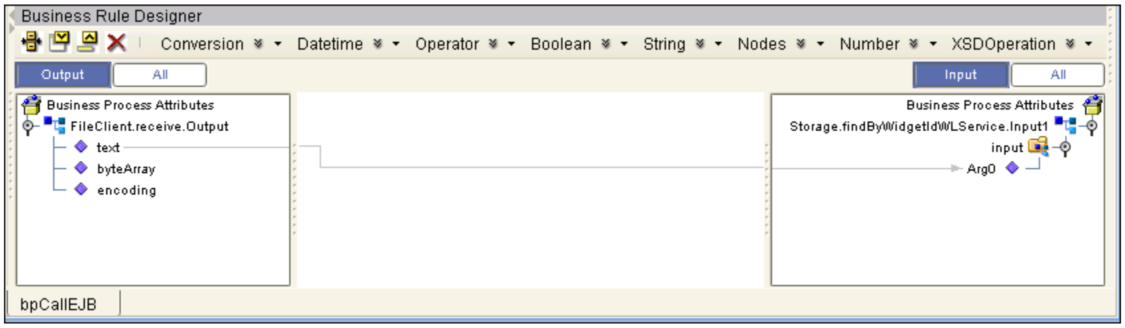
2 Create the **FileClient.receive -> Storage.findByWidgetIdWLService** Business Rule by doing the following:

A From the eInsight Business Process Designer toolbar, click the Display Business Rule Designer icon. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.

B Click on the Business Rule icon in the link between **FileClient.receive** and **Storage.findByWidgetIdWLService** to display the Business Rule's Input and Output Attributes in the Business Rule Designer.

C Map **text**, under **FileClient.receive.Output** in the Output pane of the Business Rule Designer, to **Arg0** under **Storage.findByWidgetIdWLService.Input** > **input** in the Input pane of the Business Rule Designer. To do this, click on **text** in the Output pane and drag the cursor to **Arg0** in the Input pane. A link now connects the two nodes in the Business Rule Designer (see Figure 18).

Figure 18 FileClient.receive -> Storage.findByWidgetIdWLService Rule



3 Create the **Flow -> Storage.getQuantityWLService** Business Rule by doing the following:

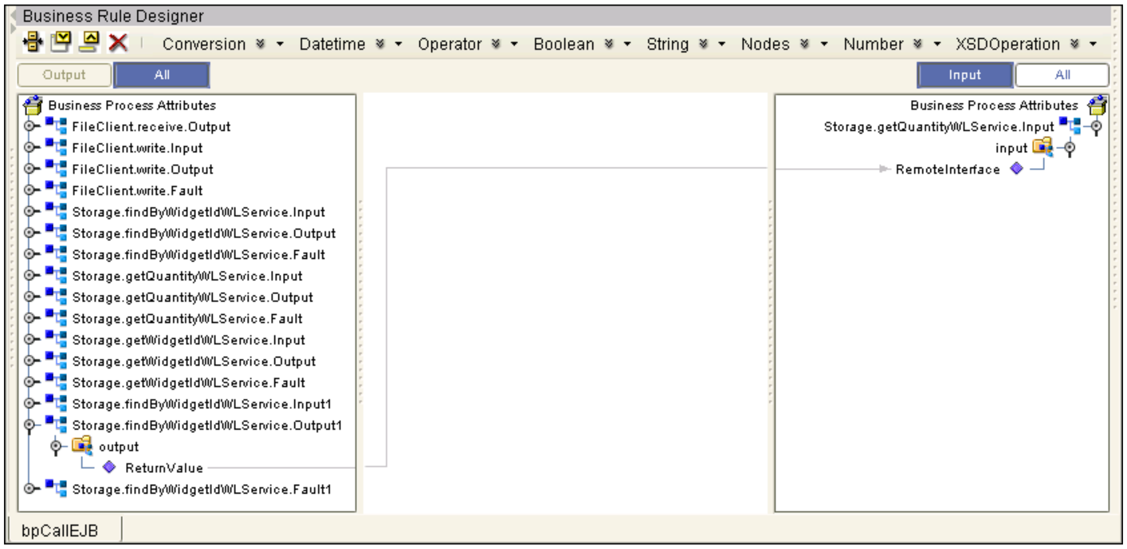
A Add a new Business Rule between the **Flow** Activity and the **Storage.getQuantityWLService** Activity. Click on the new rule.

B Double-click the new Business Rule icon to open the Business Rule Designer.

C From the Business Rule Designer, Map **ReturnValue**, under **Storage.findByWidgetIdWLService.Output1 -> output** in the Output pane of the Business Rule Designer, to **RemoteInterface** under **Storage.getQuantityWLService.Input** > **input** in the Input pane of the Business Rule Designer (see **Figure 19 on page 44**).

Note: *It is important to map the ReturnValue output of home methods (in this example, findByWidgetIdWLService) to the RemoteInterface input of the EJB method (in this case, getQuantityWLService).*

Figure 19 Flow -> Storage.getQuantityWLService Rule


Business Rule Designer
Conversion
Datetime
Operator
Boolean
String
Nodes
Number
XSDOperation
Output
All
Input
All
Business Process Attributes
FileClient.receive.Output
FileClient.write.Input
FileClient.write.Output
FileClient.write.Fault
Storage.findByWidgetIdWLService.Input
Storage.findByWidgetIdWLService.Output
Storage.findByWidgetIdWLService.Fault
Storage.getQuantityWLService.Input
Storage.getQuantityWLService.Output
Storage.getQuantityWLService.Fault
Storage.getWidgetIdWLService.Input
Storage.getWidgetIdWLService.Output
Storage.getWidgetIdWLService.Fault
Storage.findByWidgetIdWLService.Input1
Storage.findByWidgetIdWLService.Output1
output
ReturnValue
Storage.findByWidgetIdWLService.Fault1
Business Process Attributes
Storage.getQuantityWLService.Input
input
RemoteInterface
bpCallEJB

4 Create the **Flow -> Storage.getWidgetIdWLService** Business Rule by doing the following:

A Add a new Business Rule between the **Flow** Activity and the **Storage.getWidgetIdWLService** Activity.

B Double-click the new Business Rule icon to open the Business Rule Designer.

C Map **ReturnValue**, under **Storage.findByWidgetIdWLService.Output1 -> output** in the Output pane of the Business Rule Designer, to **RemoteInterface** under **Storage.getQuantityWLService.Input** > **input** in the Input pane of the Business Rule Designer.

5 Create the **Flow.end -> FileClient.write** Business Rule by doing the following:

A Add a new Business Rule between the **Flow.end** Activity and the **FileClient.write** Activity.

B Double-click the new Business Rule icon to open the Business Rule Designer.

C From the Business Rule Designer's **String** menu, select **concat**. A **concat** method box appears. Double-click the **String** field of the **concat** method box to add a String value. Enter **WidgetId:** as the String value.

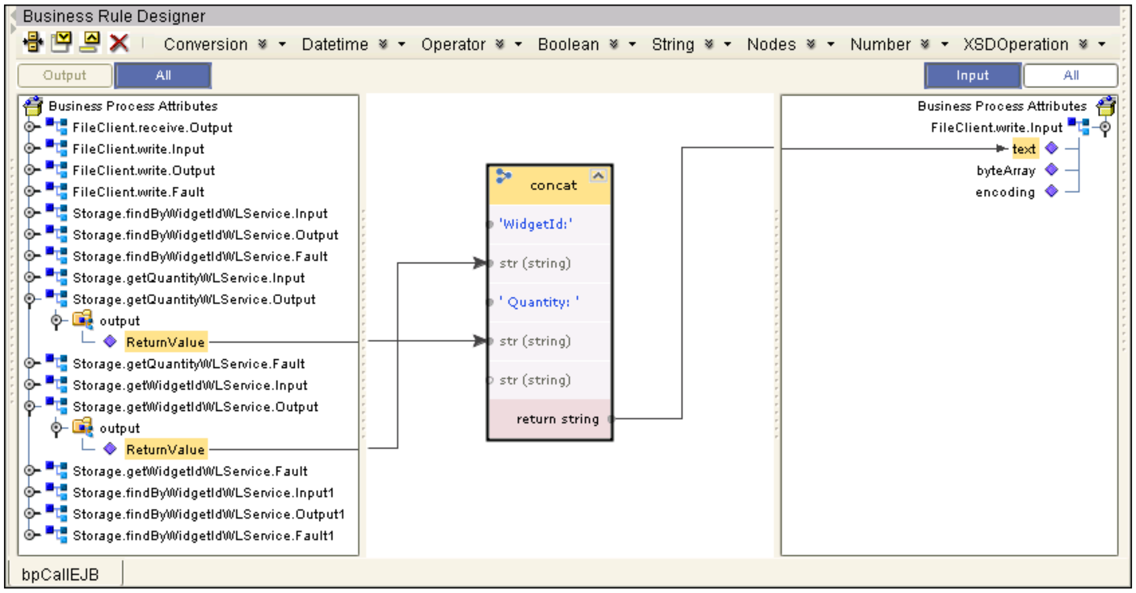
D Map **ReturnValue**, under **Storage.getWidgetIdWLService.Output -> output** in the Output pane of the Business Rule Designer, to the **str (String)** input node of the **concat** method box. An additional **str (String)** field is added to the concat method box.

E From the concat method box, double-click the bottom (unlinked) **str (String)** field. Enter **Quantity:** as the value. An additional **str (String)** field is added to the concat method box.

F Map **ReturnValue**, under **Storage.getQuantityWLService.Output -> output** in the Output pane of the Business Rule Designer, to the bottom (unlinked) **str (String)** input node of the **concat** method box.

G Map the **return string** output node of the **concat** method box, to **text**, under **FileClient.write.Input** in the Input pane of the Business Rule Designer (see Figure 20).

Figure 20 Flow.end -> FileClient.write Rule


Business Rule Designer
Conversion
Datetime
Operator
Boolean
String
Nodes
Number
XSDOperation
Output
All
Input
All
Business Process Attributes
FileClient.receive.Output
FileClient.write.Input
FileClient.write.Output
FileClient.write.Fault
Storage.findByWidgetIdWLService.Input
Storage.findByWidgetIdWLService.Output
Storage.findByWidgetIdWLService.Fault
Storage.getQuantityWLService.Input
Storage.getQuantityWLService.Output
output
ReturnValue
Storage.getQuantityWLService.Fault
Storage.getWidgetIdWLService.Input
Storage.getWidgetIdWLService.Output
output
ReturnValue
Storage.getWidgetIdWLService.Fault
Storage.findByWidgetIdWLService.Input1
Storage.findByWidgetIdWLService.Output1
Storage.findByWidgetIdWLService.Fault1
concat
'WidgetId:'
str (string)
' Quantity: '
str (string)
str (string)
return string
Business Process Attributes
FileClient.write.Input
text
byteArray
encoding
bpCallEJB

6 From the Business Process Designer toolbar, click the **Synchronize Graphical Model and Business Process** icon to synchronize the graphical interface to the Business Process code.

7 Save your changes to the Repository.

7.6.4 Creating a Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a Project's components.

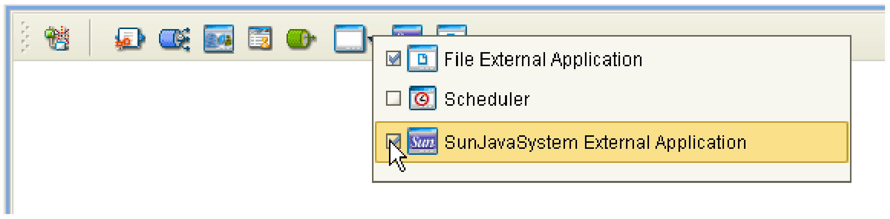
1 From the Project Explorer tree, right-click the new **prjSunAppServer_Sample_BPEL** Project and select **New > Connectivity Map** from the shortcut menu.

2 The New Connectivity Map appears and a node for the Connectivity Map is added under the Project on the Project Explorer tree labeled **CMap1**. Rename the Connectivity Map to **cmCallEJB**.

Selecting the External Applications

In the Connectivity Map, eWays are associated with External Applications. For example, to establish a connection to an external Sun Java System Application Server application, you must first select Sun Java System AppServer External Application as an External Application to use in your Connectivity Map (see Figure 21).

Figure 21 Connectivity Map - External Applications



1 Click the **External Application** icon on the Connectivity Map toolbar,

2 Select the External Applications that are necessary to create your Project (for this sample, **SunJavaSystem External Application** and **File**). Icons representing the selected External Applications are added to the Connectivity Map toolbar.

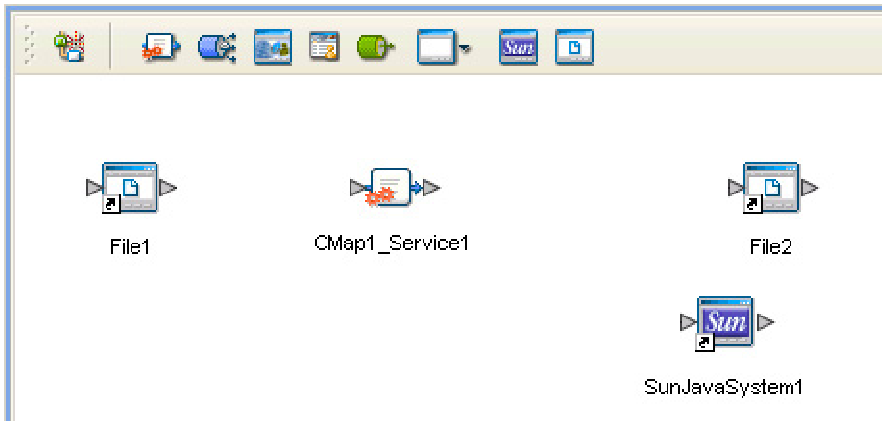
Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

1 For this sample, add the following components to the Connectivity Map canvas as displayed in Figure 22:

- File External Application (2 for this sample).
- **Service** (1 for this sample) A service is a container for Java Collaborations, Business Processes, eTL processes, and so forth.
- **Sun Java System AppServer External Application** (1 for this sample)

Figure 22 Connectivity Map with Components



2 Rename the components on the Connectivity Map as follows:

- **File1** External Application to **FileIn**
- **File2** External Application to **FileOut**
- Rename the **cmCallEJB_Service1** Service to **cmCallEJB_bpCallEJB1**

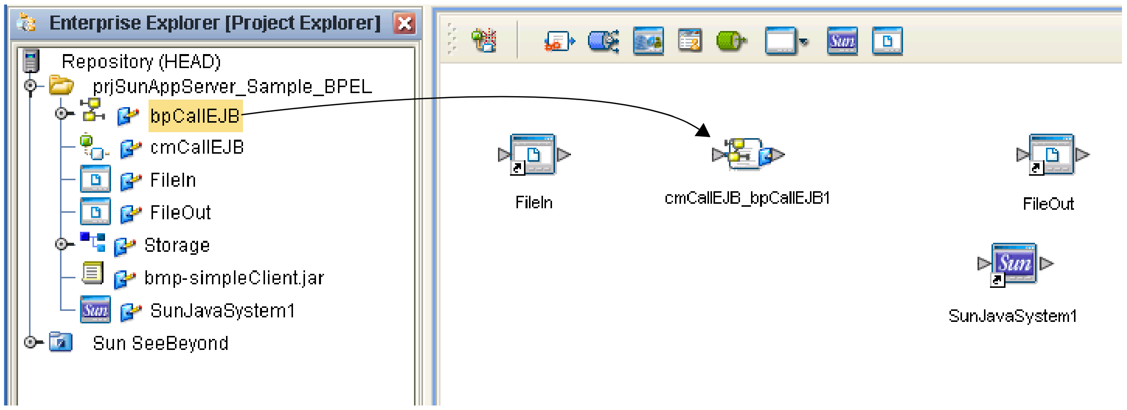
3 Save your current changes to the Repository.

7.6.5 Binding the eWay Components

Next, the components are associated and Bindings are created in the Connectivity Map.

4 Drag and drop the **bpCallEJB** Business Process from the Project Explorer tree to the **cmCallEJB_bpCallEJB1** Service. If the Business Process is successfully associated, the Service's icon changes to a Business Process icon (see Figure 23).

Figure 23 Connectivity Map - Associating the Project's Components



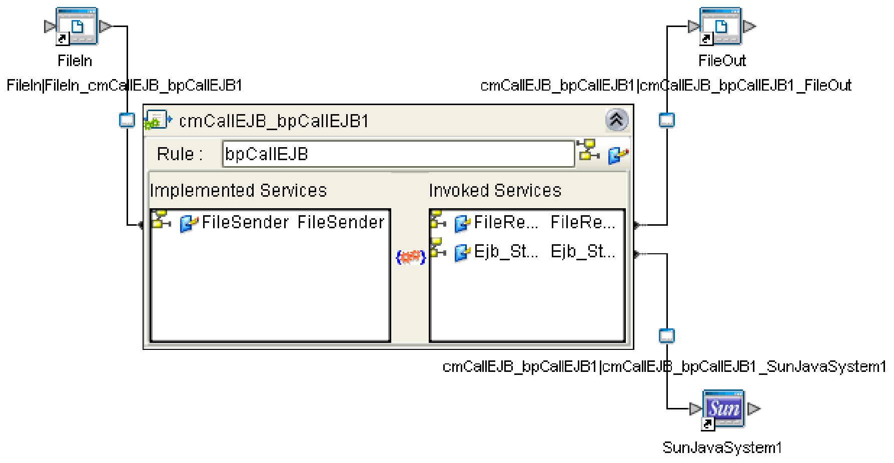
5 Double-click **cmCallEJB_bpCallEJB1**. The **cmCallEJB_bpCallEJB1** binding dialog box appears using the **bpCallEJB** Rule.

6 From the **cmCallEJB_bpCallEJB1** binding dialog box, drag **FileSender** (under Implemented Services) to the output node of the **FileIn** (File) External Application.

7 From the **cmCallEJB_bpCallEJB1** binding dialog box, drag **FileReceiver** (under Invoked Services) to the input node of the **FileOut** External Application

8 From the **cmCallEJB_bpCallEJB1** binding dialog box, drag **Ejb_Storage** (under Invoked Services) to the input node of the **SunJavaSystem1** External Application (see Figure 24).

Figure 24 Connectivity Map - Binding the Project's Components



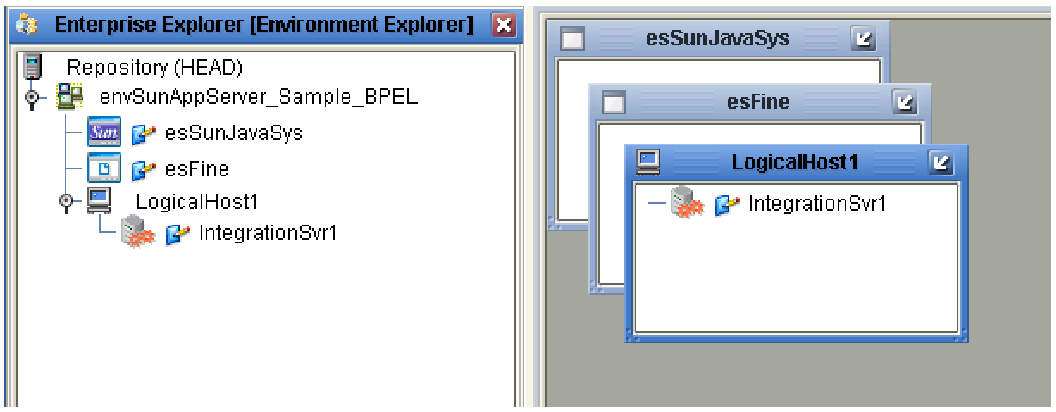
9 Save your current changes to the Repository.

7.6.6 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and message servers used by a Project and contain the configuration information for these components. Environments are created using the Environment Editor.

1. From the Enterprise Explorer, click the **Environment Explorer** tab.
2. Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.
3. Rename the new Environment to **envSunAppServer_Sample_BPEL**.
4. Right-click **envSunAppServer_Sample_BPEL** and select **New > SunJavaSystem External System**. Name the External System **esSunJavaSys**. Click **OK**. **esSunJavaSys** is added to the Environment Editor.
5. Right-click **envSunAppServer_Sample_BPEL** and select **New > File External System**. Name the External System **esFile**. Click **OK**. **esFile** is added to the Environment Editor.
6. Right-click **envSunAppServer_Sample_BPEL** and select **New > Logical Host**. **LogicalHost1** is added to the Environment Editor.
7. From the Environment Explorer tree, right-click **LogicalHost1** and select **New > Sun SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost1 (see**Figure 25 on page 48**).

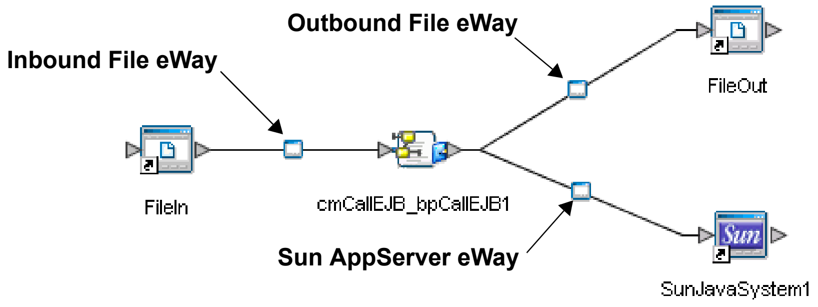
Figure 25 Environment Editor



7.6.7 Configuring the eWays

The prjSunAppServer_Sample_BPEL Project uses three eWays, each represented in the Connectivity Map as a node between an External Application and a Service (see Figure 26). eWays facilitate communication and movement of data between the external applications and the eGate system.

Figure 26 eWay Properties - Connectivity Map



Configuring the File eWay Properties

The Sun AppServer eWay properties are set in both the Connectivity Map and the Environment Explorer.

Modify the File eWay Connectivity Map Properties

1. Double-click the inbound **FileIn eWay**. The **Properties Editor** opens to the inbound File eWay properties. Modify the properties for your specific system, including the settings for the inbound File eWay in Table 5, and click **OK**.

Table 5 Inbound File eWay Settings

Inbound eWay Connectivity Map Properties	
Input file name	SunEJB_In*.txt

2. In the same way, modify the outbound File eWay properties for your system, including the settings in Table 6 on page 49, and click **OK**.

Table 6 Outbound File eWay Settings

Outbound eWay Connectivity Map Properties	
Output file name	SunEJB_out.dat

Modify the File eWay Environment Explorer Properties

1. From the **Environment Explorer** tree, right-click the File eWay External System (**esFile** in this sample), and select **Properties** from the shortcut menu. The Properties Editor appears.
2. Modify the File eWay Environment properties for your system, including the settings in Table 7, and click **OK**.

Table 7 File eWay Environment Settings

Outbound eWay Environment Properties	
Inbound File eWay > Parameter Settings	
Directory	An input directory (for example C:/temp
Outbound File eWay > Parameter Settings	
Directory	An output directory (for example C:/temp

Configuring the Sun AppServer eWay Properties

The Sun AppServer eWay properties are set in both the Connectivity Map and the Environment Explorer. For more information on the Sun AppServer eWay properties and the Properties Editor, see **"Configuring the Sun Java™ System Application Server eWay Properties" on page 22** or see the *Sun SeeBeyond eGate™ Integrator User's Guide*.

Modify the Sun AppServer eWay Connectivity Map Properties

1. From the **Connectivity Map**, double-click the **Sun AppServer eWay**. The Properties Editor opens to the Sun AppServer eWay properties.
2. Modify the Sun AppServer eWay Connectivity Map properties for your system, including the settings in Table 8, and click **OK**.

Table 8 Sun AppServer eWay Connectivity Map Properties

Sun AppServer eWay Connectivity Map Properties	
Parameter Settings Set as directed, otherwise use the default settings	
Output file name	SunEJB_out.dat

Modify the Sun AppServer eWay Environment Explorer Properties

1. From the **Environment Explorer** tree, right-click the Sun AppServer eWay External System **(esSunJavaSys** in this sample), and select **Properties** from the shortcut menu. The Properties Editor appears.
2. Modify the Sun AppServer eWay Environment properties for your system, including the settings in **Table 9 on page 50**, and click **OK**.

Table 9 Sun AppServer eWay Environment Properties

Sun AppServer eWay Environment Properties	
parameter-settings Set as directed, otherwise use the default settings	
EjbJndiName	corbaname:iiop:1.2@<*host name*>:<*port number*>#ejb/MyStorageBin

7.6.8 Configuring the Integration Server

You must set your Sun SeeBeyond Integration Server Password property before deploying your Project.

1. From the Environment Explorer, right-click **IntegrationSvr1** under your **Logical Host**, and select **Properties** from the shortcut menu. The Integration Server Properties Editor appears.
2. Click the **Password** property field under **Sun SeeBeyond Integration Server Configuration**. An ellipsis appears in the property field.

3 Click the ellipsis. The **Password Settings** dialog box appears. Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.

4 Click **OK** to accept the new property and close the Properties Editor.

For more information on deploying a Project see the *Sun SeeBeyond Java™ Composite Application Platform Suite Deployment Guide*.

7.6.9 Creating the Deployment Profile

Deployment Profiles are specific instances of a Project in a particular Environment. A Deployment Profile is created using the Enterprise Designer's Deployment Editor.

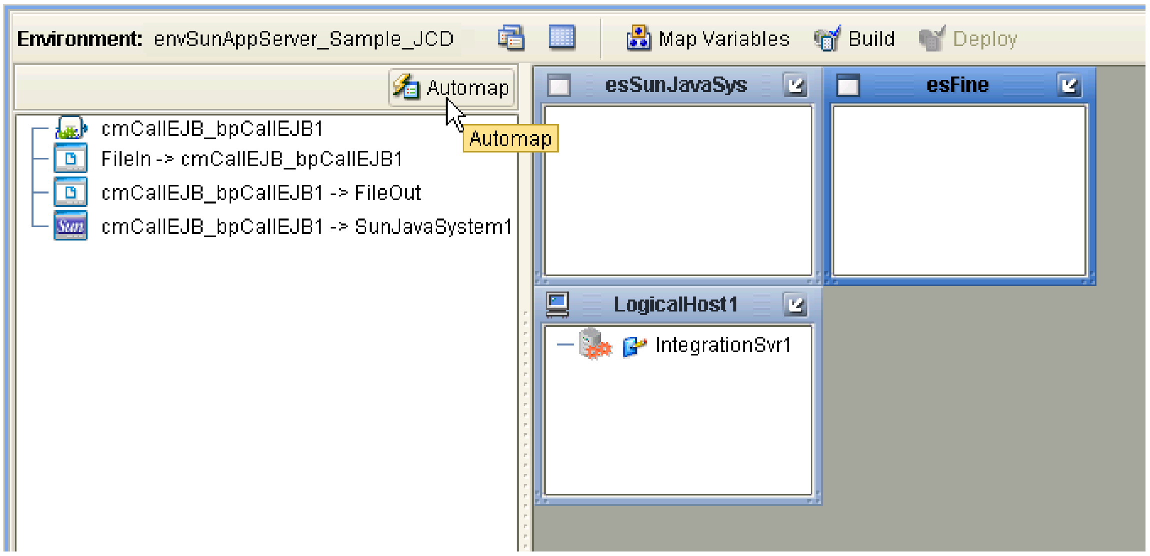
To create a Deployment Profile, do the following:

1 From the Enterprise Explorer's Project Explorer, right-click the Project and select **New** > **Deployment Profile**.

2 From the **Create Deployment Profile** dialog box, enter a name for the Deployment Profile (for this example, **dpSunAppServer_Sample_BPEL**). Select the appropriate Environment (**envSunAppServer_Sample_BPEL)** and click **OK**.

3 Click the **Auto Map** icon as displayed in Figure 27.

Figure 27 Deployment Profile - Auto Map

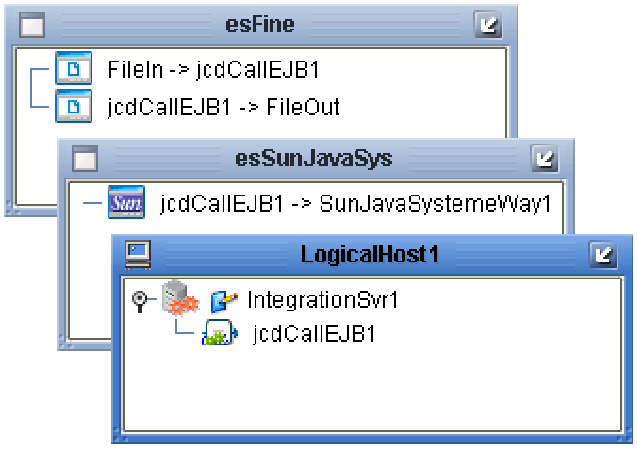


Environment: envSunAppServer_Sample_JCD

cmCallEJB_bpCallEJB1
FileIn -> cmCallEJB_bpCallEJB1
cmCallEJB_bpCallEJB1 -> FileOut
cmCallEJB_bpCallEJB1 -> SunJavaSystem1

4 The Project's components are automatically mapped to their system windows as seen in **Figure 28 on page 52**.

Figure 28 Deployment Profile



5 Click **Activate**. When activation succeeds, save the changes to the Repository.

7.6.10 Creating and Starting the Domain

To deploy your Project, you must first create a domain. A domain is an instance of a Logical Host.

Create and Start the Domain

1 Navigate to your ***<JavaCAPS51>*\logicalhost** directory (where <*JavaCAPS51*> is the location of your Sun Java Composite Application Platform Suite installation.

2 Double-click the **domainmgr.bat** file. The **Domain Manager** appears.

3 If you have already created a domain, select your domain in the Domain Manager and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

4 If there are no existing domains, a dialog box indicates that you can create a domain now. Click **Yes**. The **Create Domain** dialog box appears.

5 Make any necessary changes to the **Create Domain** dialog box and click **Create**. The new domain is added to the Domain Manager. Select the domain and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

6 For more information about creating and managing domains see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

7.6.11 Building and Deploying the Project

The Build process compiles and validates the Project's Java files and creates the Project EAR file.

Note: *If your EJB was compiled with JDK 1.5, see* **"Build the Deployment (EAR) File Using the Commandline Codegen Tool" on page 53**.

Build the Project

1. From the Deployment Editor toolbar, click the **Build** icon.
2. If there are any validation errors, a **Validation Errors** pane will appear at the bottom of the Deployment Editor and displays information regarding the errors. Make any necessary corrections and click **Build** again.
3. After the Build has succeeded you are ready to deploy your Project.

Deploy the Project

1. From the Deployment Editor toolbar, click the **Deploy** icon. Click **Yes** when the **Deploy** prompt appears.
2. A message appears when the project is successfully deployed.

7.6.12 Build the Deployment (EAR) File Using the Commandline Codegen Tool

The Sun SeeBeyond Enterprise Designer is packaged with **JDK 1.4**. If your EJB is compiled with **JDK version 1.5 or above**, you must use **Command Line Codegen** to build the Deployment (EAR) file. Please refer to the *Sun SeeBeyond eGate Integrator User's Guide (Building an Application File From the Command Line)* for details.

For a simple description of how to use the Commandline Codegen Tool to create an EAR file, do the follow:

1. Download the CommandLineCodegen tool from the Sun Java Composite Application Platform Suite Installer.
2. Extract the files into a local directory.
3. Download and install **JDK 1.5** from Sun website, or make a note of the JDK that was used to compile the EJB.
4. Download and install **ANT** (version 1.6.2 or above) from Apache Software Foundation at **http://ant.apache.org**.
5. Change the **build.properties** file in the commandlinecodegen directory. At a minimum, you must change the following parameters (with sample data). Make sure there is no trailing space after each parameter. The parameters below are usually different for each specific environment.

```
commandline.rep.url=http://localhost:12000/rep
commandline.rep.user=Administrator
commandline.rep.pass=STC
commandline.rep.dir=localrepository
commandline.rep.projectName=prjSunAppeWay
commandline.rep.projectDeployName=dpBmpSimple
```

6. Open a Command Prompt to the **commandlinecodegen** directory.
7. Make sure **JAVA_HOME** and **ANT_HOME** are pointing to the proper directory (this should be consistent with step 3 and 4 above).
8. Run commandline codegen by issuing the following command from the Command Prompt in your commandlinecodegen directory:

```
ant -propertyfile build.properties
```

9 The resulting EAR file is be located in the following directory:

```
commandlinecodegen\localrepository\DEST
```

This is where the EAR file for the next step is located.

10 Deploy the EAR file into Sun SeeBeyond eGate Integrator using the Sun SeeBeyond Enterprise Manager. See the Sun SeeBeyond eGate™ System Administration Guide (Chapter 3) for details.

7.6.13 Running the Project

To run your deployed sample Project do the following

1 From your configured input directory, paste (or rename) the sample input file to trigger the eWay.

2 From your output directory, verify the output data.

Chapter 8

Implementing a Project Using Java Collaboration Definitions (JCD)

This chapter provides an introduction to the Sun AppServer eWay components and information on how these components are created and implemented in an eGate Project. It is assumed that the reader understands the basics of creating a Project using the Sun SeeBeyond Enterprise Designer. For more information on creating an eGate Project see the *Sun SeeBeyond eGate™ Tutorial* and the *Sun SeeBeyond eGate™ Integrator User's Guide*.

What's in This Chapter

8.1 Sun AppServer eWay Components

This chapter presents a sample Sun Java System AppServer eWay Project created using the same procedures as the sample end-to-end Project provided in the *eGate Integrator Tutorial*. The eWay components that are unique to the Sun AppServer eWay include the following:

Sun AppServer eWay Properties File

The properties configuration file for the Sun AppServer eWay contains the properties that are used to connect with a specific external system. These parameters are set using the Properties Editor. For more information about the Sun AppServer eWay properties file and the Properties Editor see **"Configuring the Sun Java™ System Application Server eWay Properties" on page 22**.

Sun Java System AppServer OTD Wizard

The **Sun Java System AppServer OTD Wizard** builds an Object Type Definition (OTD) from an EJB or JAR file. The wizard generates Object Type Definitions that map input and output message segments at the field level.

8.2 The Sun AppServer eWay JCE Sample Project

This following pages provide directions for creating the **prjSunAppServer_Sample_JCD** Project The **prjSunAppServer_Sample_JCD** Project demonstrates how the Sun AppServer eWay uses Java Collaborations to provide the Business Logic. The **prjSunAppServer_Sample_JCD** Project can be downloaded from the Sun Composite Application Platform Suite Installer and imported in a nearly-complete state.

Sample Overview

The **prjSunAppServer_Sample_JCD** Project demonstrates the following:

- The inbound File eWay subscribes to an external input directory. When a target message is present the File eWay picks up the message that contains an item number (777) and publishes the message to the jcdCallEJB Java Collaboration.
- The Collaboration uses the Sun Java System Application Server eWay to query the Sun Application Server External System for the item's ID, location, and quantity, which it takes from the PointBase database provided with the Sun Java System Application Server Sample Bundle. The Collaboration writes this information to a message and publishes it to the outbound File eWay.
- The outbound File eWay publishes the new message to an output directory.

8.3 Importing a Sample Project

Sample eWay Projects are included as part of the installation package. To import a sample eWay Project to the Enterprise Designer do the following:

1. The sample files are uploaded with the eWay's documentation SAR file and downloaded from the Sun Java Composite Application Platform Suite Installer's **Documentation** tab. The **Sun_AppServer_eWay_Sample.zip** file contains the various sample Project ZIP files. Extract the samples to a local file.
2. Save all unsaved work before importing a Project.
3. From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import** from the shortcut menu. The **Import Manager** appears.
4. Browse to the directory that contains the sample Project zip file. Select the sample file (for this sample, **SunAppServer_Sample_BPEL.zip**) and click **Import**. After the sample Project is successfully imported, click **Close**.
5. Before an imported sample Project can be run you must do the following:
   - Create an **Environment** (see **"Creating an Environment" on page 67**)
   - Configure the eWays for your specific system (see **"Configuring the eWays" on page 68**)
   - Create a **Deployment Profile** (see **"Creating the Deployment Profile" on page 71**)

- Create and start a domain (see **"Creating and Starting the Domain" on page 72**)
- Build and deploy the Project (see **"Building and Deploying the Project" on page 72**)

8.4 Creating the prjSunAppServer_Sample_JCD Project

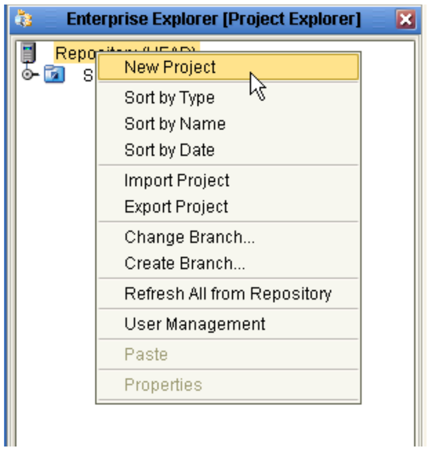
The following pages provide step by step directions for manually creating the sample Project's components.

8.4.1 Creating a Project

The first step is to create a new Project in the SeeBeyond Enterprise Designer.

1 From the Enterprise Designer's Project Explorer tree, right-click the Repository and select **New Project** (see Figure 29). A new Project (**Project1**) appears on the Project Explorer tree.

Figure 29 Enterprise Explorer - New Project



2 Rename the Project to **prjSunAppServer_Sample_JCD**.

8.4.2 Create an Object Type Definition Using the OTD Wizard

The Sun Java System AppServer OTD Wizard creates an Object Type Definition from a deployed EJB or JAR file. This sample uses file the file **bmp-simpleClient.jar**, provided with the Sun Java System Application Server Samples Bundle. Make sure that this sample bundle is downloaded prior to manually creating the sample.

For step by step directions on creating the Storage OTD used in this sample using the Sun Java System AppServer OTD Wizard, see **"Create an Object Type Definition Using the OTD Wizard" on page 38**. Once you have created the Storage OTD, proceed to **"Creating Collaboration Definitions" on page 58**.

8.4.3 Creating Collaboration Definitions

The next step in the sample is to create the **jcdCallEJB** Java Collaboration using the Collaboration Definition Wizard (Java). Once the Collaboration has been created, the Collaboration's Business Rules can be written using the Collaboration Editor (Java).

Creating the jcdCallEJB Collaboration Definition

The **jcdCallEJB** Collaboration defines transactions from the inbound File eWay to the Sun AppServer eWay and the outbound File eWay.

1 From the Project Explorer, right-click the sample Project and select **New > Collaboration Editor (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

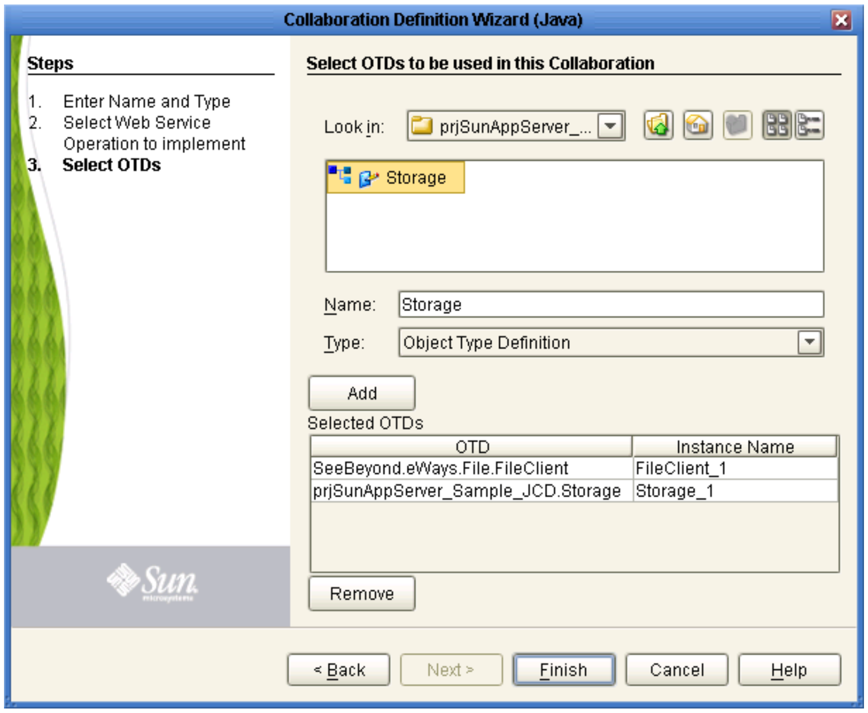
2 Enter a Collaboration Definition name (for this sample **jcdCallEJB**) and click **Next**.

3 For Step 2 or the wizard, from the Web Services Interfaces selection window, double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive.** Click **Next**.

4 For Step 3 of the wizard, from the Select OTDs selection window, double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **FileClient_1** OTD is added to the Selected OTDs field.

5 Click **Up One Level** to return to the Repository. From the Select OTDs selection window, double-click **prjSunAppServer_Sample_JCD** > **Storage**. The **Storage OTD** is added to the Selected OTDs field (see Figure 30).

Figure 30 Collaboration Definition Wizard (Java) - Select Web Service


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
Select OTDs to be used in this Collaboration
Look in: prjSunAppServer_...
Storage
Add
Remove
< Back
Next >
Finish
Cancel
Help

Name: Storage
Type: Object Type Definition

Selected OTDs

OTD	Instance Name
SeeBeyond.eWays.File.FileClient	FileClient_1
prjSunAppServer_Sample_JCD.Storage	Storage_1

6 Click **Finish**. The Collaboration Editor (Java) with the new **jcdCallEJB** Collaboration appears in the right pane of the Enterprise Designer.

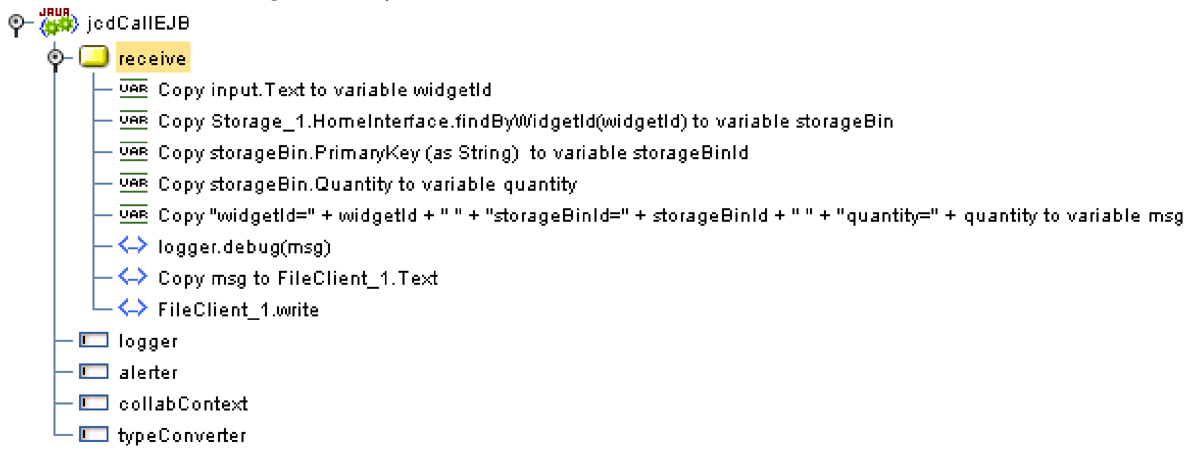
8.4.4 Using the Collaboration Editor (Java)

The next step in the sample is to create the Business Rules of the Java Collaboration using the Collaboration Editor (Java).

Create the jcdCallEJB Business Rules

The **jcdCallEJB** Collaboration contains the Business Rules displayed in Figure 31.

Figure 31 jcdCallEJB Collaboration Business Rules



```
jcdCallEJB
  receive
    Copy input.Text to variable widgetId
    Copy Storage_1.HomeInterface.findByWidgetId(widgetId) to variable storageBin
    Copy storageBin.PrimaryKey (as String)  to variable storageBinId
    Copy storageBin.Quantity to variable quantity
    Copy "widgetId=" + widgetId + " " + "storageBinId=" + storageBinId + " " + "quantity=" + quantity to variable msg
    logger.debug(msg)
    Copy msg to FileClient_1.Text
    FileClient_1.write
  logger
  alerter
  collabContext
  typeConverter
```

To create the **jcdCallEJB** Collaboration Business Rules do the following:

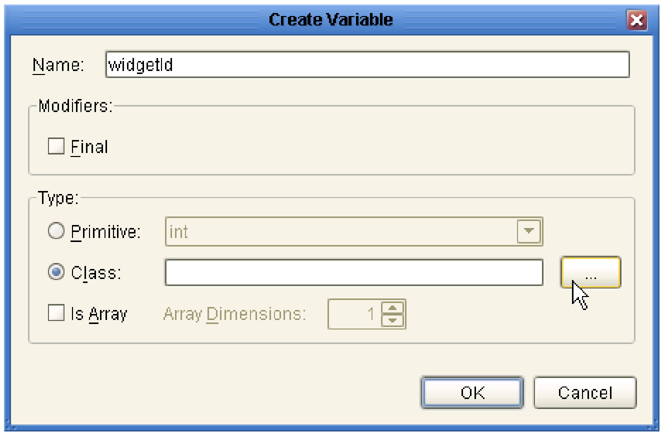
1. If the Collaboration Editor is not open, double-click **jcdCallEJB** in the Project Explorer tree to open Collaboration Editor to the Sun **jcdCallEJB** Collaboration.

2. To create comments for the Business Rules, click the comment icon on the Business Rules toolbar. The **Enter a Comment** dialog box appears. Enter the comment and click **OK**. The comment is placed on the Business Rules tree under the last selected item. You can move (drag) the new comment up or down the Business Rules tree.

3. Create the **Copy input.Text to variable widgetId** (variable) rule:

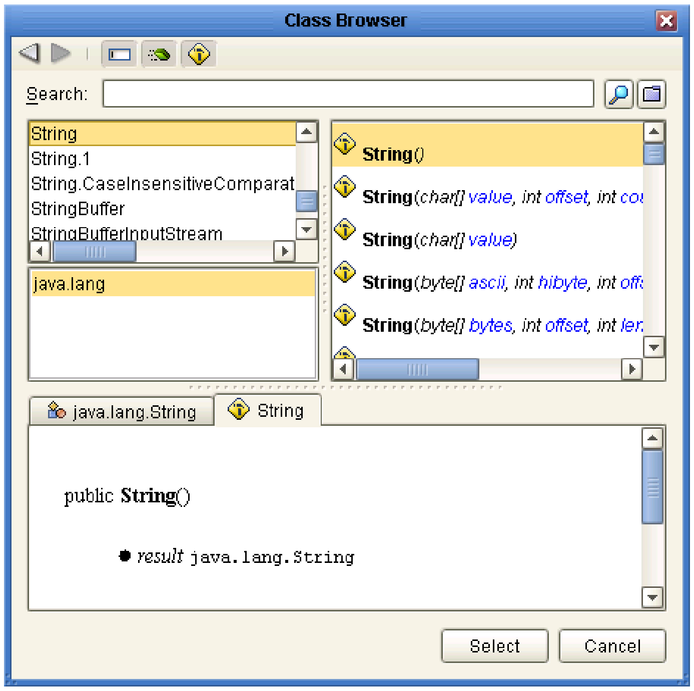
   A. From the Business Rules toolbar, click **Local Variable**. The **Create Variable** dialog box appears (see Figure 32).

Figure 32 jcdCallEJB Collaboration Business Rules - Create Variable



B From the **Create Variable** dialog box, enter **widgetId** as the variable name. For **Type**, select **Class** and click the ellipsis (...) button. The **Class Browser** dialog box appears (see Figure 33).

Figure 33 jcdCallEJB Collaboration Business Rules - Class Browser


Class Browser
Search:
String
String.1
String.CaseInsensitiveComparat
StringBuffer
StringBufferInputStream
java.lang
String()
String(char[] value, int offset, int cou
String(char[] value)
String(byte[] ascii, int hibyte, int offs
String(byte[] bytes, int offset, int len
java.lang.String
String
Select
Cancel

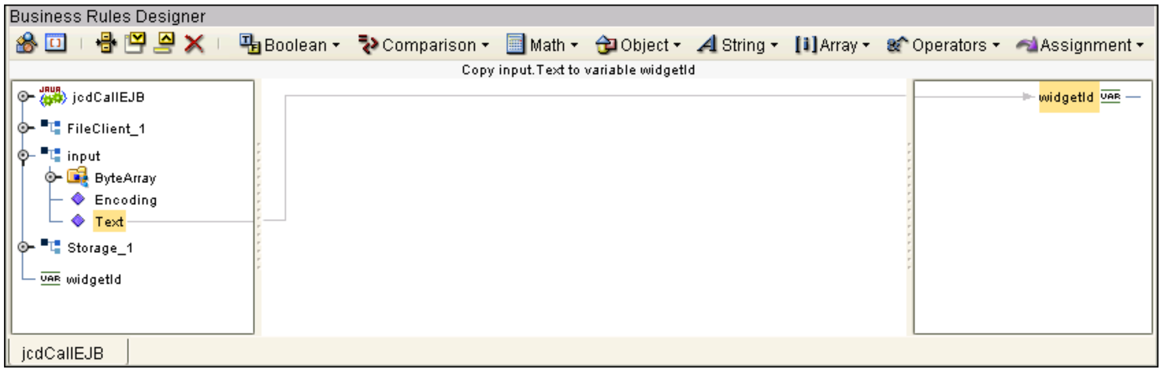
```
public String()

    • result java.lang.String
```

C From the **Class Browser**, select **String** as the class. The method defaults to the String() constructor. Click **Select**.

D Click **OK** to close the **Create Variable** dialog box and create your variable.

E Map **Text** under **input** in the left pane of the Business Rules Designer, to **widgetId** (variable) in the right pane of the Business Rules Designer. To do this, click on **Text** in the left pane of the Business Rules Designer, and drag your cursor to **widgetId** (variable) in the right pane of the Business Rules Designer. A link now connects the two nodes (see Figure 34).

Figure 34 Java Collaboration Editor - jcdCallEJB Business Rules



4 Create the **Copy Storage_1.HomeInterface.findByWidgetId(widgetId) to variable storageBin** rule:

A From the Business Rules toolbar, click **Local Variable**. The **Create Variable** dialog box appears.

B From the **Create Variable** dialog box, enter **storageBin** as the variable name. For **Type**, select **Class** and click the ellipsis (...) button. The **Class Browser** dialog box appears.

C Select **storageBin** as the class, and select **getEJBHome()** as the method. Click **Select**.

D Click **OK** to close the **Create Variable** dialog box and create your variable.

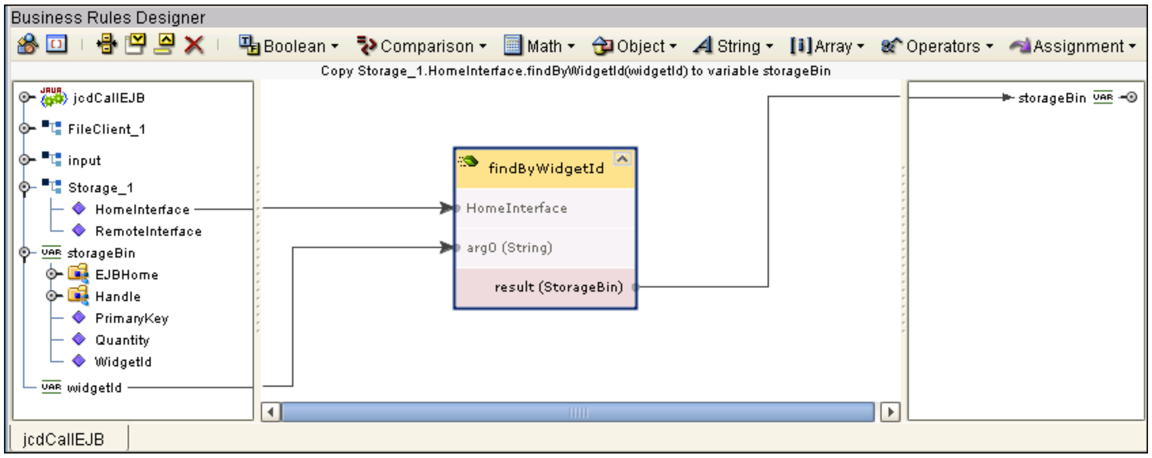
E Right-click **HomeInterface** under **Storage_1** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.

F Select **findByWidgetId(String widgetId)** from the method selection window. The **findByWidgetId** method box appears.

G Map **widgetId** in the left pane of the Business Rules Designer, to the **arg0 (String)** input node of the **findByWidgetId** method box.

H Map the **result (storageBin)** output node of the **findByWidgetId** method box, to **storageBin** (variable) in the right pane of the Business Rules Designer (see Figure 35).

Figure 35 Java Collaboration Editor - jcdCallEJB Business Rules



Copy Storage_1.HomeInterface.findByWidgetId(widgetId) to variable storageBin

5 Create the **Copy storageBin.PrimaryKey (as String) to variable storageBinId** rule:

A From the Business Rules toolbar, click **Local Variable**. The **Create Variable** dialog box appears.

B From the **Create Variable** dialog box, enter **storageBinId** as the variable name. For **Type**, select **Class** and click the ellipsis (...) button. The **Class Browser** dialog box appears.

C From the **Class Browser**, select **String** as the class. The method defaults to the String() constructor. Click **Select**.

D Click **OK** to close the **Create Variable** dialog box and create your variable.

E From the Business Rules Designer toolbar's **Object** menu, select **Cast**. The **Cast** dialog box appears.

F From the **Cast** dialog box, select **Class** and click the ellipsis (...) button. The **Class Browser** dialog box appears.

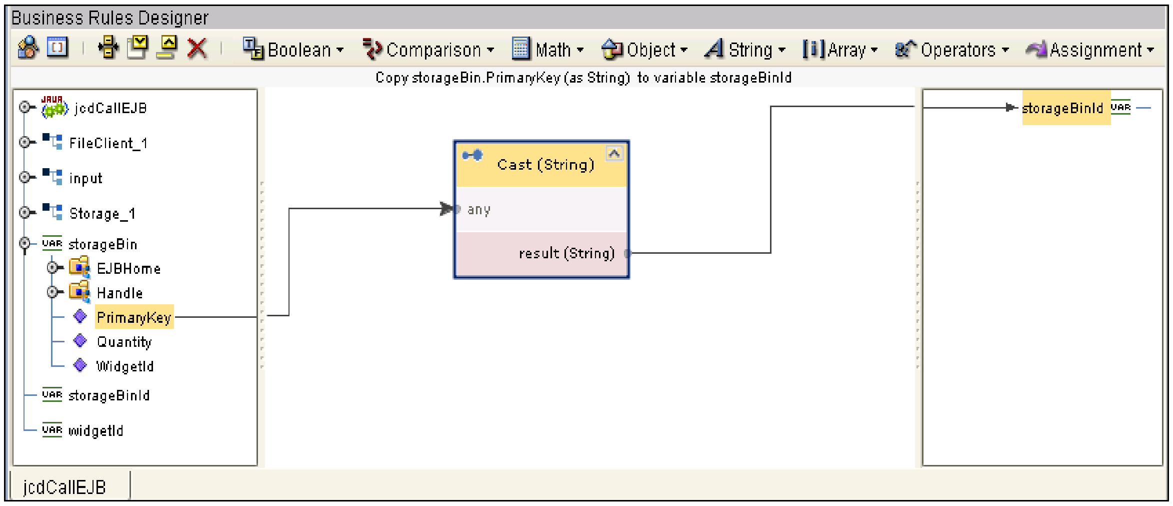
G From the **Class Browser**, select **String** as the class. The method defaults to the String() constructor. Click **Select**.

H Click **OK** to close the **Cast** dialog box. The **Cast (String)** method box appears.

I Map **PrimaryKey** under **storageBin** (variable) in the left pane of the Business Rules Designer, to the **any** input node of the **Cast (String)** method box.

J Map the **result (String)** output node of the **Cast (String)** method box, to **storageBinId** in the right pane of the Business Rules Designer (see Figure 36).

Figure 36 Java Collaboration Editor - jcdCallEJB Business Rules



6 Create the **Copy storageBin.Quantity to variable quantity** rule:

A From the Business Rules toolbar, click **Local Variable**. The **Create Variable** dialog box appears.

B From the **Create Variable** dialog box, enter **quantity** as the variable name. For Type, click **Primitive:** and select **int**. Click **OK**

C Map **Quantity** under **storageBin** (variable) in the left pane of the Business Rules Designer, to **quantity** in the right pane of the Business Rules Designer.

7 Create the **Copy "widgetId=" + widgetId + " " + "storageBinId=" + storageBinId + " " + "quantity" + quantity to variable msg** rule:

A From the Business Rules toolbar, click **Local Variable**. The **Create Variable** dialog box appears.

B From the **Create Variable** dialog box, enter **msg** as the variable name.For Type, select **Class** and click the ellipsis (...) button. The **Class Browser** dialog box appears.

C From the **Class Browser**, select **String** as the class. The method defaults to the String() constructor. Click **Select**.

D From the Business Rules Designer toolbar's **String** menu, select **Add**. The **Add** method box appears.

E Double-click the **value1** field of the **Add** method box. Enter **widgetId=** as the value.

F Map **widgetId** (variable) in the left pane of the Business Rules Designer, to the **value2** input node of the **Add** method box. The **value3** field is added to the **Add** method box.

G Double-click the **value3** field of the **Add** method box. Leaving the value blank, click anywhere on the Business Rules Designer canvas to save the blank value and add the **value4** field to the **Add** method box.

H Double-click the **value4** field of the **Add** method box. Enter **storageBinId=** as the value. The **value5** field is added to the **Add** method box.

I Map **storageBinId** in the left pane of the Business Rules Designer, to the **value5** input node of the **Add** method box. The **value6** field is added to the **Add** method box.

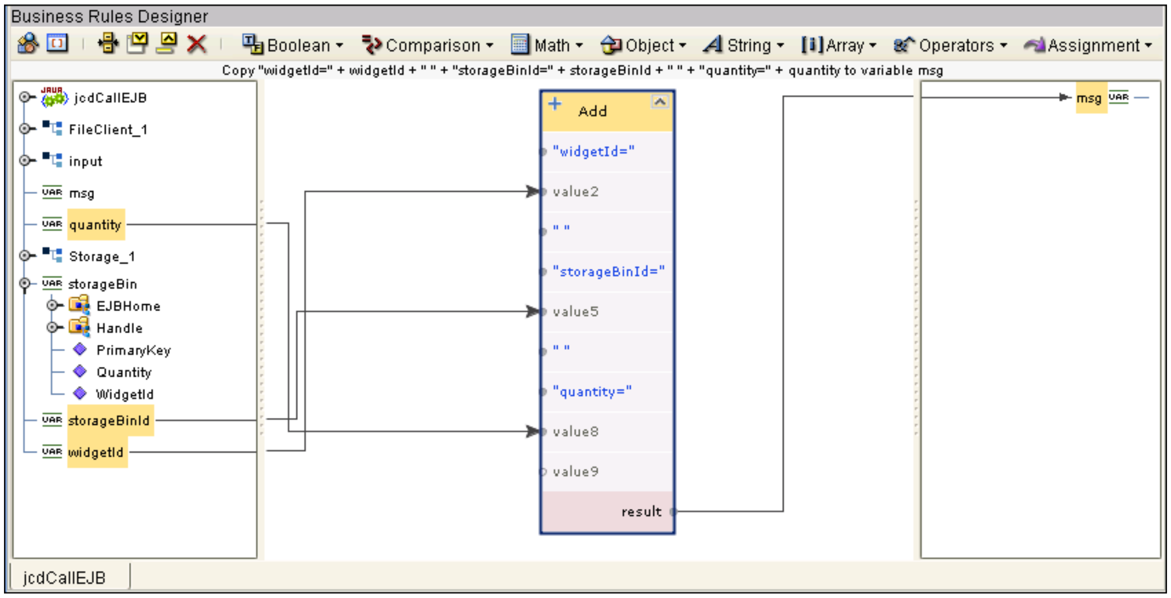
J Double-click the **value6** field of the **Add** method box. Leave the value blank and click anywhere on the Business Rules Designer canvas to save the blank value and add the **value7** field to the **Add** method box.

K Double-click the **value7** field of the **Add** method box. Enter **quantity=** as the value. The **value8** field is added to the **Add** method box.

L Map **quantity** in the left pane of the Business Rules Designer, to the **value8** input node of the **Add** method box.

M Map the **result** output node of the **Add** method box to **msg** (variable) in the right pane of the Business Rules Designer (see Figure 37).

Figure 37 Java Collaboration Editor - jcdCallEJB Business Rules



Copy "widgetId=" + widgetId + " " + "storageBinId=" + storageBinId + " " + "quantity=" + quantity to variable msg

8 Create the **logger.debug(msg)** rule:

A From the Business Rules toolbar, click the **rule** icon to add a new rule.

B Right-click **logger** (the logger field) under **jcdCallEJB** in the left pane of the Business Rules Designer, and select **Select method to call** from the shortcut menu.

C Select **debug (Object arg0)** from the method selection window. The **debug** method box appears.

D Map **msg** in the left pane of the Business Rules Designer, to the **arg0(Object)** input node of the debug method box.

9 Create the **Copy msg to FileClient_1.Text** rule:

A From the Business Rules toolbar, click the **rule** icon to add a new rule.

B. Map **msg** in the left pane of the Business Rules Designer, to **Text** under **FileClient_1** in the right pane of the Business Rules Designer.

10. Create the **FileClient_1.write** rule:

    A. From the Business Rules toolbar, click the **rule** icon to add a new rule.

    B. Right-click **FileClient_1** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.

    C. Select **write()** from the method selection window. The **write** method box appears.

11. From the editor's toolbar, click **Validate** to check the Collaboration for errors.

12. Save your current changes to the repository.

For more information on how to create Business Rules using the Collaboration Editor see the *Sun SeeBeyond eGate™ Integrator User's Guide*.

8.4.5 Creating a Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a Project's components.

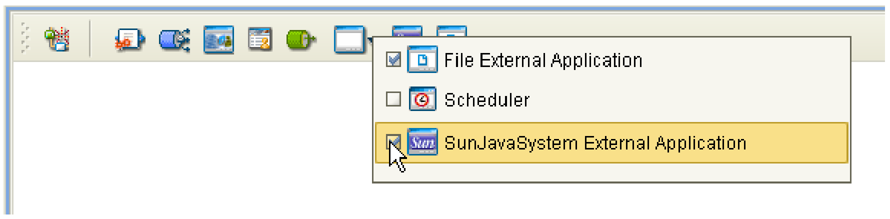
1. From the Project Explorer tree, right-click the new **prjSunAppServer_Sample_JCD** Project and select **New > Connectivity Map** from the shortcut menu.

2. The New Connectivity Map appears and a node for the Connectivity Map is added under the Project on the Project Explorer tree labeled **CMap1**. Rename the Connectivity Map to **cmCallEJB**.

Selecting the External Applications

In the Connectivity Map, eWays are associated with External Applications. For example, to establish a connection to an external Sun Java System Application Server application, you must first select Sun Java System AppServer External Application as an External Application to use in your Connectivity Map (see Figure 38).

Figure 38 Connectivity Map - External Applications



1. Click the **External Application** icon on the Connectivity Map toolbar,

2. Select the External Applications that are necessary to create your Project (for this sample, **SunJavaSystem External Application** and **File**). Icons representing the selected External Applications are added to the Connectivity Map toolbar.

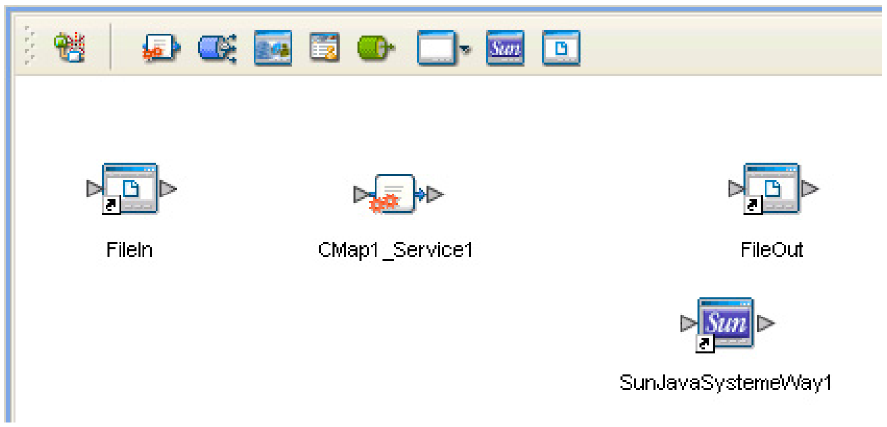
Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

1 For this sample, add the following components to the Connectivity Map canvas as displayed in Figure 39:

- File External Application (2 for this sample).
- **Service** (1 for this sample) A service is a container for Java Collaborations, Business Processes, eTL processes, and so forth.
- **Sun Java System AppServer External Application** (1 for this sample)

Figure 39 Connectivity Map with Components



2 Rename the components on the Connectivity Map as follows:

- **File1** External Application to **FileIn**
- **File2** External Application to **FileOut**
- Rename the **cmCallEJB_Service1** Service to **jcdCallEJB1**

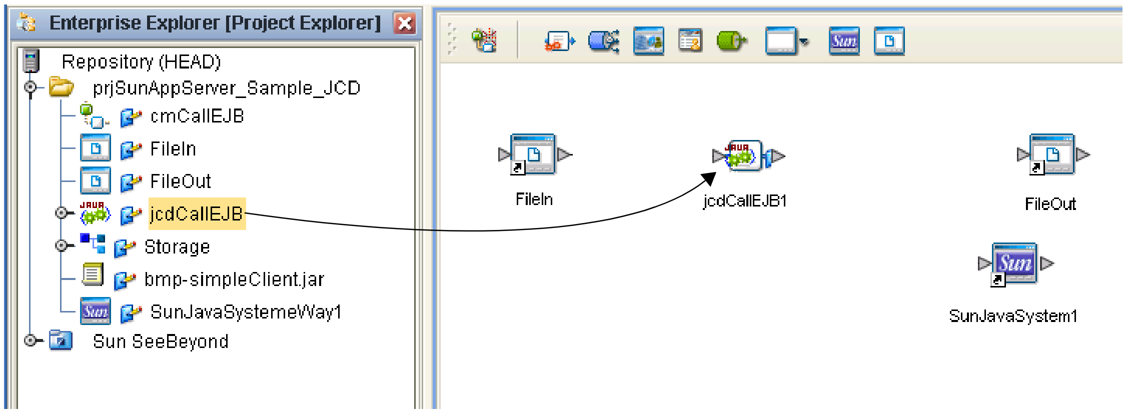
3 Save your current changes to the Repository.

8.4.6 Binding the eWay Components

Next, the components are associated and Bindings are created in the Connectivity Map.

4 Drag and drop the **jcdCallEJB** Collaboration from the Project Explorer tree to the **jcdCallEJB1** Service. If the Collaboration is successfully associated, the Service’s icon changes to a Collaboration icon (see Figure 40).

Figure 40 Connectivity Map - Associating the Project's Components



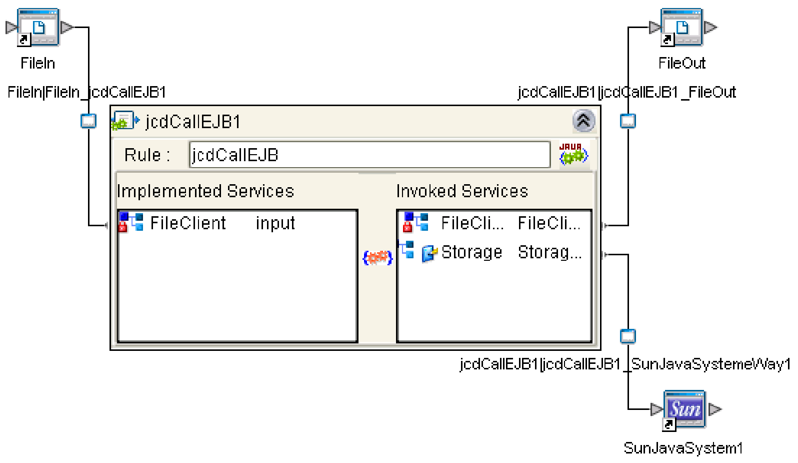
5 Double-click **jcdCallEJB1**. The **jcdCallEJB1** binding dialog box appears using the **jcdCallEJB** Rule.

6 From the **jcdCallEJB1** binding dialog box, drag **FileClient Input** (under Implemented Services) to the output node of the **FileIn** (File) External Application. A link now appears between the jcdCallEJB1 binding dialog box and the **FileIn** eWay.

7 From the **jcdCallEJB1** binding dialog box, drag **FileClient_1** (under Invoked Services) to the input node of the **FileOut** External Application

8 From the **jcdCallEJB1** dialog box, drag **Storage_1** (under Invoked Services) to the input node of the **SunJavaSystemeWay1** External Application (see Figure 41).

Figure 41 Connectivity Map - Binding the Project's Components



9 Save your current changes to the Repository.

8.4.7 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and message servers used by a Project and contain the configuration information for these components. Environments are created using the Environment Editor.

1 From the Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **envSunAppServer_Sample_JCD**.

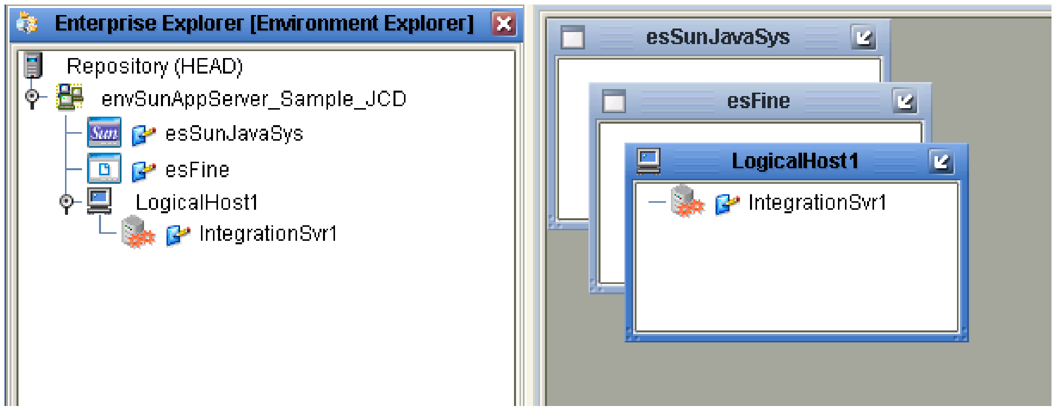
4 Right-click **envSunAppServer_Sample_JCD** and select **New > SunJavaSystem External System**. Name the External System **esSunJavaSys**. Click **OK**. **esSunJavaSys** is added to the Environment Editor.

5 Right-click **envSunAppServer_Sample_JCD** and select **New > File External System**. Name the External System **esFile**. Click **OK**. **esFile** is added to the Environment Editor.

6 Right-click **envSunAppServer_Sample_JCD** and select **New > Logical Host**. **LogicalHost1** is added to the Environment Editor.

7 From the Environment Explorer tree, right-click **LogicalHost1** and select **New > Sun SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost1 (seeFigure 42).

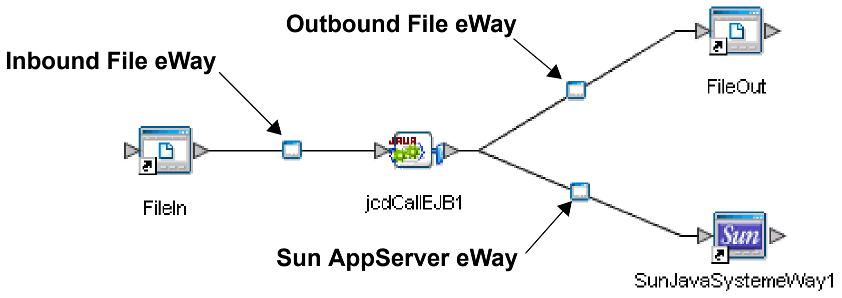
Figure 42 Environment Editor



8.4.8 Configuring the eWays

The prjSunAppServer_Sample_JCD Project uses three eWays, each represented in the Connectivity Map as a node between an External Application and a Service (see **Figure 43 on page 69**). eWays facilitate communication and movement of data between the external applications and the eGate system.

Figure 43 eWay Properties - Connectivity Map



Configuring the File eWay Properties

The Sun AppServer eWay properties are set in both the Connectivity Map and the Environment Explorer.

Modify the File eWay Connectivity Map Properties

1 Double-click the inbound **FileIn eWay**. The **Properties Editor** opens to the inbound File eWay properties. Modify the properties for your specific system, including the settings for the inbound File eWay in Table 10, and click **OK**.

Table 10 Inbound File eWay Settings

Inbound eWay Connectivity Map Properties	
Input file name	SunEJB_In*.txt

2 In the same way, modify the outbound File eWay properties for your system, including the settings in Table 11, and click **OK**.

Table 11 Outbound File eWay Settings

Outbound eWay Connectivity Map Properties	
Output file name	SunEJB_out.dat

Modify the File eWay Environment Explorer Properties

1 From the **Environment Explorer** tree, right-click the File eWay External System (**esFile** in this sample), and select **Properties** from the shortcut menu. The Properties Editor appears.

2 Modify the File eWay Environment properties for your system, including the settings in Table 12, and click **OK**.

Table 12 File eWay Environment Settings

Outbound eWay Environment Properties	
Inbound File eWay > Parameter Settings	
Directory	An input directory (for example C:/temp
Outbound File eWay > Parameter Settings	
Directory	An output directory (for example C:/temp

Configuring the Sun AppServer eWay Properties

The Sun AppServer eWay properties are set in both the Connectivity Map and the Environment Explorer. For more information on the Sun AppServer eWay properties and the Properties Editor, see **"Configuring the Sun Java™ System Application Server eWay Properties" on page 22** or see the *Sun SeeBeyond eGate™ Integrator User's Guide*.

Modify the Sun AppServer eWay Connectivity Map Properties

1. From the **Connectivity Map**, double-click the **Sun AppServer eWay**. The Properties Editor opens to the Sun AppServer eWay properties.
2. Modify the Sun AppServer eWay Connectivity Map properties for your system, including the settings in Table 13, and click **OK**.

Table 13 Sun AppServer eWay Connectivity Map Properties

Sun AppServer eWay Connectivity Map Properties	
Parameter Settings Set as directed, otherwise use the default settings	
Output file name	SunEJB_out.dat

Modify the Sun AppServer eWay Environment Explorer Properties

1. From the **Environment Explorer** tree, right-click the Sun AppServer eWay External System **(esSunJavaSys** in this sample), and select **Properties** from the shortcut menu. The Properties Editor appears.
2. Modify the Sun AppServer eWay Environment properties for your system, including the settings in Table 14, and click **OK**.

Table 14 Sun AppServer eWay Environment Properties

Sun AppServer eWay Environment Properties	
parameter-settings Set as directed, otherwise use the default settings	
EjbJndiName	corbaname:iiop:1.2@<*host name*>:<*port number*>#ejb/MyStorageBin

8.4.9 Configuring the Integration Server

You must set your Sun SeeBeyond Integration Server Password property before deploying your Project.

1. From the Environment Explorer, right-click **IntegrationSvr1** under your **Logical Host**, and select **Properties** from the shortcut menu. The Integration Server Properties Editor appears.
2. Click the **Password** property field under **Sun SeeBeyond Integration Server Configuration**. An ellipsis appears in the property field.

3 Click the ellipsis. The **Password Settings** dialog box appears. Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.

4 Click **OK** to accept the new property and close the Properties Editor.

For more information on deploying a Project see the *Sun SeeBeyond Java™ Composite Application Platform Suite Deployment Guide*.

8.4.10 Creating the Deployment Profile

Deployment Profiles are specific instances of a Project in a particular Environment. A Deployment Profile is created using the Enterprise Designer's Deployment Editor.

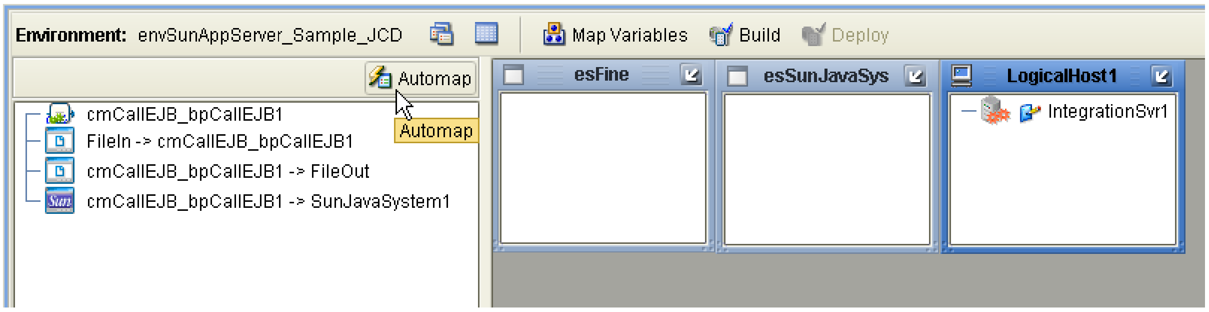
To create a Deployment Profile, do the following:

1 From the Enterprise Explorer's Project Explorer, right-click the Project and select **New** > **Deployment Profile**.

2 From the **Create Deployment Profile** dialog box, enter a name for the Deployment Profile (for this example, **dpSunAppServer_Sample_JCD**). Select the appropriate Environment (**envSunAppServer_Sample_JCD)** and click **OK**.

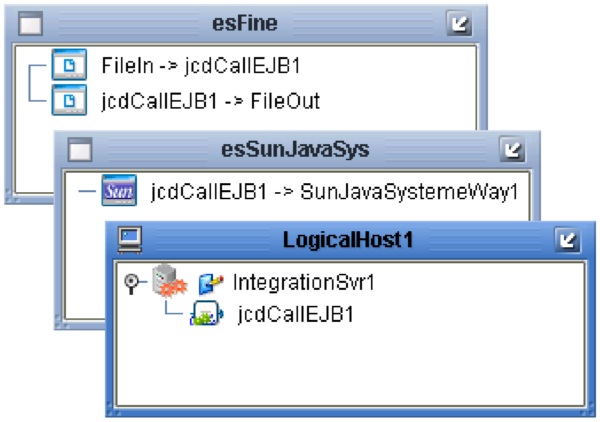
3 Click the **Auto Map** icon as displayed in Figure 44.

Figure 44 Deployment Profile - Auto Map


Environment: envSunAppServer_Sample_JCD
Map Variables
Build
Deploy
Automap
cmCallEJB_bpCallEJB1
FileIn -> cmCallEJB_bpCallEJB1
cmCallEJB_bpCallEJB1 -> FileOut
cmCallEJB_bpCallEJB1 -> SunJavaSystem1
Automap
esFine
esSunJavaSys
LogicalHost1
IntegrationSvr1

4 The Project's components are automatically mapped to their system windows as seen in Figure 45.

Figure 45 Deployment Profile



5 Click **Activate**. When activation succeeds, save the changes to the Repository.

8.4.11 Creating and Starting the Domain

To deploy your Project, you must first create a domain. A domain is an instance of a Logical Host.

Create and Start the Domain

1. Navigate to your ***<JavaCAPS51>*\logicalhost** directory (where <*JavaCAPS51*> is the location of your Sun Java Composite Application Platform Suite installation.
2. Double-click the **domainmgr.bat** file. The **Domain Manager** appears.
3. If you have already created a domain, select your domain in the Domain Manager and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.
4. If there are no existing domains, a dialog box indicates that you can create a domain now. Click **Yes**. The **Create Domain** dialog box appears.
5. Make any necessary changes to the **Create Domain** dialog box and click **Create**. The new domain is added to the Domain Manager. Select the domain and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.
6. For more information about creating and managing domains see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

8.4.12 Building and Deploying the Project

The Build process compiles and validates the Project's Java files and creates the Project EAR file.

The Sun SeeBeyond Enterprise Designer is packaged with **JDK 1.4**. If your EJB is compiled with **JDK version 1.5 or above**, you must use **Command Line Codegen** to build the Deployment (EAR) file. See **"Build the Deployment (EAR) File Using the Commandline Codegen Tool" on page 53**.

Build the Project

1. From the Deployment Editor toolbar, click the **Build** icon.
2. If there are any validation errors, a **Validation Errors** pane will appear at the bottom of the Deployment Editor and displays information regarding the errors. Make any necessary corrections and click **Build** again.
3. After the Build has succeeded you are ready to deploy your Project.

Deploy the Project

1. From the Deployment Editor toolbar, click the **Deploy** icon. Click **Yes** when the **Deploy** prompt appears.
2. A message appears when the project is successfully deployed.

8.4.13 Running the Project

To run your deployed sample Project do the following

1. From your configured input directory, paste (or rename) the sample input file to trigger the eWay.
2. From your output directory, verify the output data.

Index