SeeBeyond ICAN Suite

Sun Java System Application Server eWay Intelligent Adapter User's Guide

Release 5.0

Version 20041202110403.

Contents

Chapter 1

Introducing the Sun Java System Application Server eWay

This document describes how to install, configure, and implement the Sun™ Java™ System Application Server eWay Intelligent Adapter, in a typical ICAN environment.

This chapter provides a brief overview of operations and components, general features, and system requirements of the Sun Java System Application Server eWay Intelligent Adapter (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 Java System Application Server eWay Intelligent Adapter

SeeBeyond's Sun Java System Application Server eWay enables the SeeBeyond Integrated Composite Application Network™ (ICAN™) Suite to perform the routing, processing, and caching of messages containing relevant data between Sun Application Server enabled applications and other diverse applications supported by ICAN eWay products.

1.3 About This Document

This section provides a brief outline of this user's guide.

1.3.1. Organization of Information

The Sun Java System AppServer eWay Intelligent Adapter 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 AppServer 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 included with the eWay installation.
- **Chapter 8 "Implementing a Project Using the Collaboration Editor (Java)"** 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

1.3.2. Scope of the Document

This user's guide provides a description of the Sun Java System Application Server eWay Intelligent Adapter. 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.

1.3.3. Intended Audience

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

1.3.4. Document Conventions

The following conventions are observed throughout this document.

Table 1 Document Conventions

Text	Convention	Example
Names of buttons, files, icons, parameters, variables, methods, menus, and objects	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments, code samples	`Fixed` font. Variables are shown in ***bold italic***.	`bootstrap -p` ***password***
Hypertext links	**Blue** text	See **"Document Conventions" on page 9**
Hypertext links for Web addresses (URLs) or email addresses	**Blue underlined** text	**http://www.seebeyond.com** **docfeedback@seebeyond.com**

1.4 SeeBeyond Web Site

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

http://www.seebeyond.com

1.5 SeeBeyond Documentation Feedback

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

docfeedback@seebeyond.com

Chapter 2

Installing the Sun Java System Application Server eWay

This Chapter describes how to install the Sun Java System Application Server eWay Intelligent Adapter, as well as the accompanying documentation and sample projects. It also includes the Sun AppServer eWay's supported operating systems and system requirements.

What's in This Chapter

2.1 Supported Operating Systems

The Sun Java System Application Server eWay is available for the following operating systems:

- Windows 2000, Windows XP, and Windows Server 2003
- HP Tru64 V5.1A
- HP-UX 11.0, 11i (PA-RISC), and 11i v2.0 (11.23)
- IBM AIX 5.1L and 5.2
- Red Hat Enterprise Linux AS 2.1 (Intel x86)
- Red Hat Linux 8 (Intel x86)
- Sun Solaris 8 and 9

2.2 System Requirements

The system requirements for the Sun Java System Application Server eWay are the same as those for eGate Integrator. For information, refer to the *SeeBeyond ICAN Suite Installation Guide*. It is also helpful to review the **ICAN Suite Readme.txt**, located on the installation CD-ROM, for any additional requirements prior to installation.

An eWay specific Readme file, the **Sun Java System Application Server eWay Readme**, provides the most up to date information for the Sun AppServer eWay. The Readme is uploaded along with other eWay documentation and samples, and accessed from the Enterprise Manager. See **Installing the Sun Java System Application Server eWay** on page 12, for more information on uploading and accessing the Readme.

Although the Sun Java System Application Server eWay, the Repository, and Logical Hosts run on the platforms listed under Supported Operating Systems, the Enterprise Designer requires the Windows operating system. The Enterprise Manager can run on any platform that supports Internet Explorer 6.0 and later.

2.2.1. External System Requirements

The Sun AppServer eWay supports **Sun Java System Application Server 8.0**, and requires an existing Sun Java System Application Server-compliant application or component.

2.3 Installing the Sun Java System Application Server eWay

During the eGate Integrator installation process, the Enterprise Manager, a web-based application, is used to select and upload eWays from the eGate installation CD-ROM to the Repository.

Installing the Sun Java System Application Server eWay on an eGate Supported System

The Sun Java System Application Server eWay can be installed during or after the installation of the ICAN Suite. The ICAN Suite installation process includes the following operations:

- Install the eGate Repository
- Upload products to the Repository
- Download components (including the eGate Enterprise Designer and Logical Host)

Follow the directions for installing the ICAN Suite in the *SeeBeyond ICAN Suite Installation Guide.* After you have installed eGate and other purchased core products, do the following:

1. From the Enterprise Manager's **ADMIN** tab, browse to the **Add-ons** directory and select the **ProductsManifest.xml**, and click **Submit**. The available Add-on product list is now displayed.
2. Browse to and select the following files located in the **Add-ons** directory:
   - **SunJavaSystemeWay.sar** (to install the Sun AppServer eWay)
   - **FileeWay.sar** (to install the File eWay, used with the sample project)

3 Click on the Manifest File field's **Browse** option, browse to the Add-ons **Documentation** directory, select the **ProductsManifest.xml**, and click **Submit**. The available Add-on documentation list is now displayed.

4 From the **Documentation** directory, select and upload the following file:

- **SunJavaSystemeWayDoc.sar** (to upload the Sun Java System Application Server eWay User's Guide, Javadoc, Readme, and sample projects to the Enterprise Manager)

5 Continue installing the eGate Integrator as instructed in the SeeBeyond *ICAN Suite Installation Guide.*

Adding the eWay to an Existing ICAN Suite Installation

If you are installing the eWay to an existing ICAN installation, do the following:

1 Complete steps 1 through 5 above.

2 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. 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. Click **Finish.**

7 When prompted, restart the IDE to complete the installation.

After Installation

Once the eWay is installed and configured it must then be incorporated into a project before it can perform its intended functions. See the *eGate Integrator User's Guide* for more information on incorporating the eWay into an eGate project.

The eWay's User Guide, Javadoc, Readme, and sample projects, can be accessed from the Enterprise Manager's Documentation tab.

Chapter 3

Sun Application Server Components

This section provides an overview of the various Sun Microsystem 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 to 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.

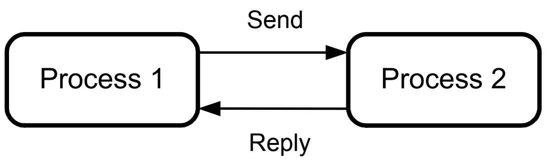
What's in This Chapter

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 18.

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.

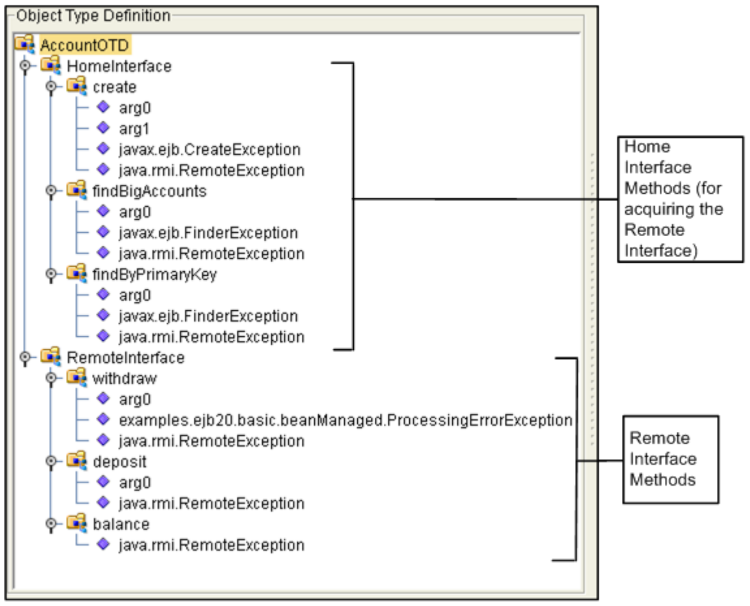
The OTD is divided into two portions:

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

  As an example, the home interface method **findBigAccounts()**, seen in Figure 2, could use the argument "balanceGreaterThan (100,000)" to find all account EJBs with a 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 AppServer 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.

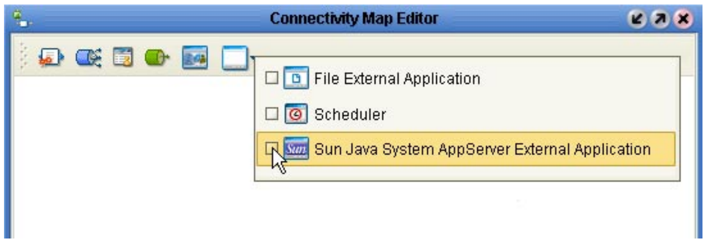
5.1.1 Selecting Sun Application Server as the External Application

To create a Sun AppServer eWay you must first create an 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 20**). The selected Sun AppServer External Application icon appears on the Connectivity Map toolbar.

Figure 3 External Applications Selection Menu

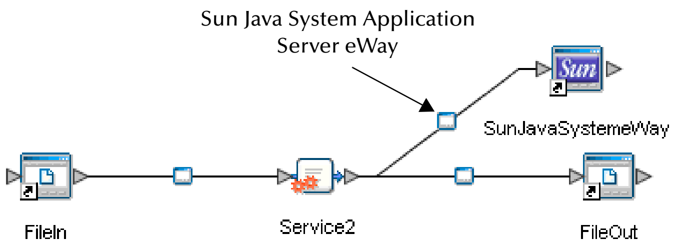

Connectivity Map Editor
File External Application
Scheduler
Sun Java System AppServer External Application

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 AppServer 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.

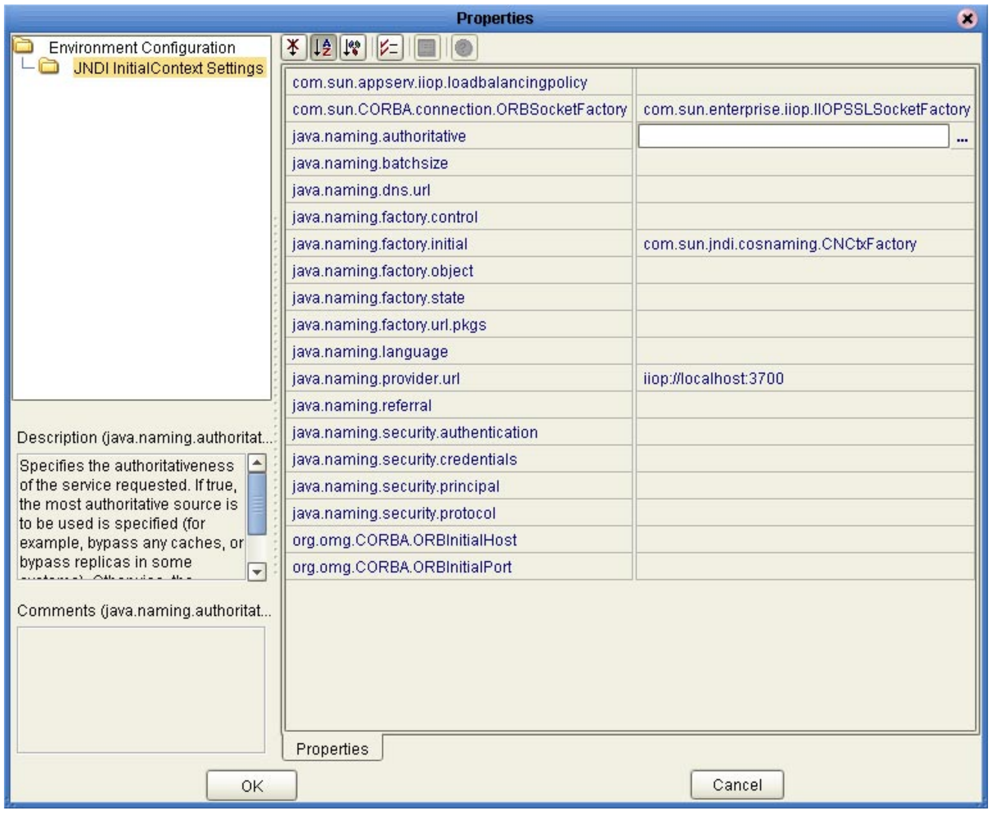
5.1.4. Using the Properties Editor

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 **JNDI InitialContext Settings** subdirectory displays the editable parameters in the right pane, as shown in Figure 5

Figure 5 Properties Editor: Sun AppServer eWay



com.sun.appserv.iiop.loadbalancingpolicy	
com.sun.CORBA.connection.ORBSocketFactory	com.sun.enterprise.iiop.IIOPSSLSocketFactory
java.naming.authoritative	
java.naming.batchsize	
java.naming.dns.url	
java.naming.factory.control	
java.naming.factory.initial	com.sun.jndi.cosnaming.CNCtxFactory
java.naming.factory.object	
java.naming.factory.state	
java.naming.factory.url.pkgs	
java.naming.language	
java.naming.provider.url	iiop://localhost:3700
java.naming.referral	
java.naming.security.authentication	
java.naming.security.credentials	
java.naming.security.principal	
java.naming.security.protocol	
org.omg.CORBA.ORBInitialHost	
org.omg.CORBA.ORBInitialPort	

Description (java.naming.authoritat...

Specifies the authoritativeness of the service requested. If true, the most authoritative source is to be used is specified (for example, bypass any caches, or bypass replicas in some

Comments (java.naming.authoritat...

2 Click on any property field to make it editable. For example, click on the **java.naming.authoritative** 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 Sun AppServer eWay Connectivity Map Properties

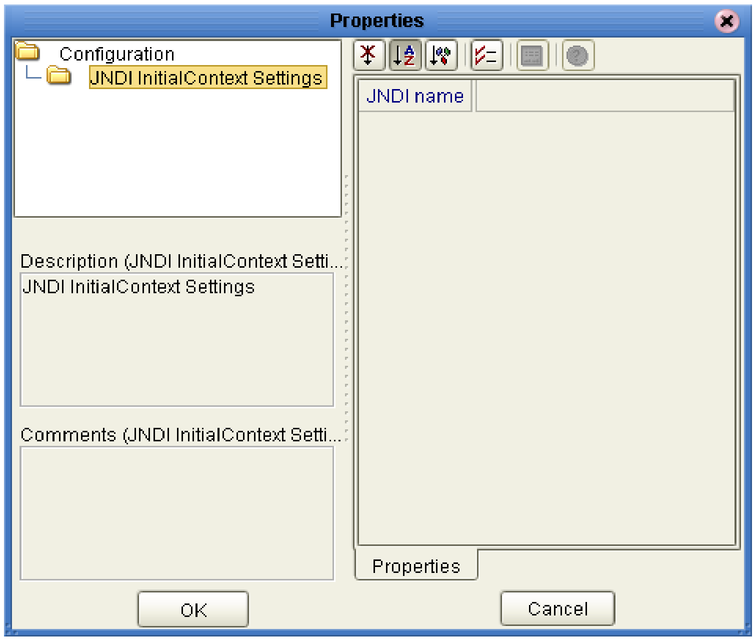
The eWay properties, accessed from the Connectivity Map, are organized into the following sections:

JNDI InitialContext Settings on page 22

5.2.1. JNDI InitialContext Settings

The **JNDI InitialContext Settings** section contains the top level properties displayed in Figure 6.

Figure 6 Properties Editor - Connectivity Map, JNDI InitialContext Settings


Properties
Configuration
JNDI InitialContext Settings
Description (JNDI InitialContext Setti...
JNDI InitialContext Settings
Comments (JNDI InitialContext Setti...
JNDI name
Properties
OK
Cancel

JNDI name

Description

Specifies a JNDI name for initial context lookup to overwrite the default JNDI name provided at the creation of the OTD if so desired, otherwise this can be left blank.

Required Value

Enter a JNDI name (String).

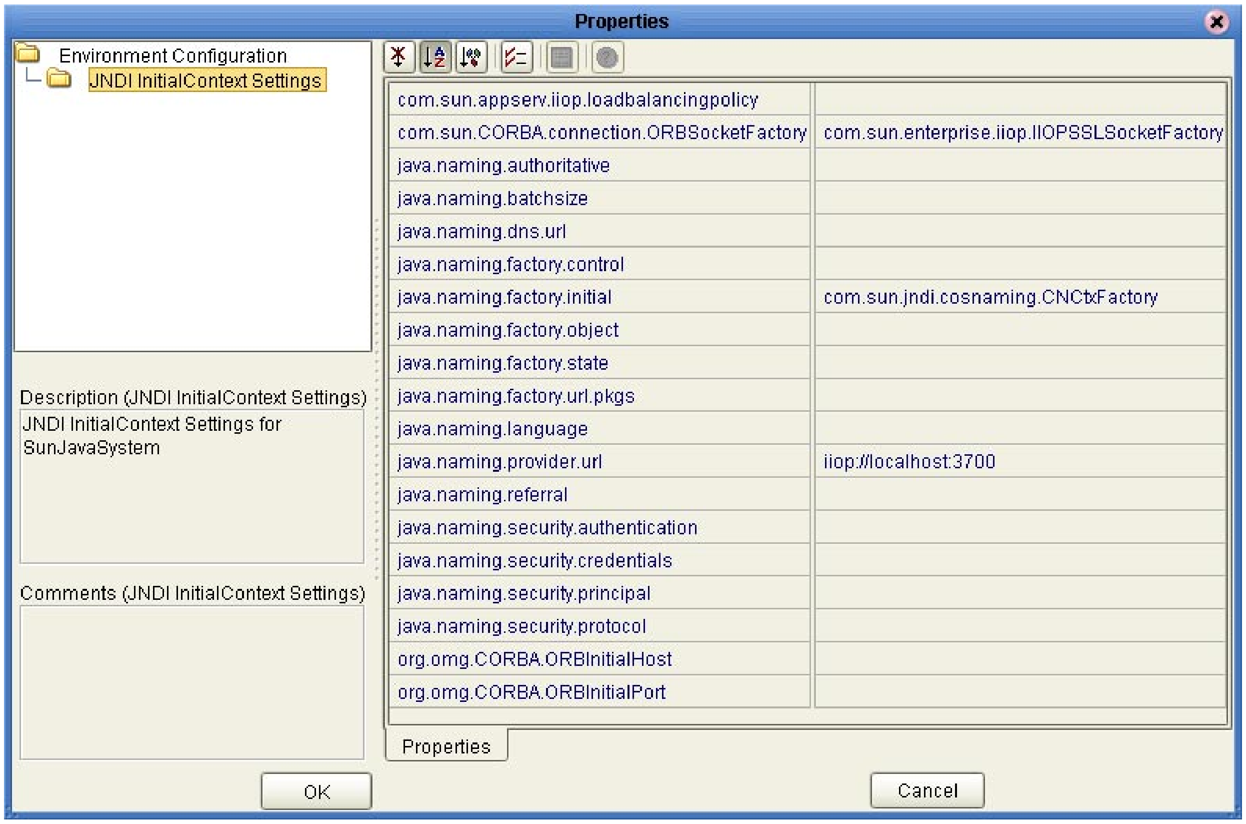
5.3 Sun AppServer eWay Environment Properties

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

5.3.1. JNDI InitialContext Settings

The **JNDI InitialContext Settings** section contains the top level properties displayed in Figure 7.

Figure 7 Environment Properties - JNDI InitialContext Settings



Description (JNDI InitialContext Settings)
JNDI InitialContext Settings for SunJavaSystem

Comments (JNDI InitialContext Settings)

com.sun.appserv.iiop.loadbalancingpolicy	
com.sun.CORBA.connection.ORBSocketFactory	com.sun.enterprise.iiop.IIOPSSLSocketFactory
java.naming.authoritative	
java.naming.batchsize	
java.naming.dns.url	
java.naming.factory.control	
java.naming.factory.initial	com.sun.jndi.cosnaming.CNCtxFactory
java.naming.factory.object	
java.naming.factory.state	
java.naming.factory.url.pkgs	
java.naming.language	
java.naming.provider.url	iiop://localhost:3700
java.naming.referral	
java.naming.security.authentication	
java.naming.security.credentials	
java.naming.security.principal	
java.naming.security.protocol	
org.omg.CORBA.ORBInitialHost	
org.omg.CORBA.ORBInitialPort	

com.sun.appserv.iiop.loadbalancingpolicy

Description

Specifies the system property used by the S1ASCtxFactory class to implement client-side load balancing if the effective name service hostname does not map to multiple IP addresses (see **java.naming.factory.initial** on page 25).

- **com.sun.appserv.iiop.loadbalancingpolicy=roundrobin,host1:port1,host2:port2...** used to implement a round robin algorithm.
- **com.sun.appserv.iiop.loadbalancingpolicy=application,host1:port1,host2:port2...** used if the round robin policy is not required.

This property provides a list of **host:port** combinations to round robin the ORBs. These host names may also map to multiple IP addresses. If you use this property along with **org.omg.CORBA.ORBInitialHost** and **org.omg.CORBA.ORBInitialPort** as system properties, the round robin algorithm will round robin across all the values provided.

Required Value

See Description.

com.sun.CORBA.connection.ORBSocketFactory

Description

Specifies variable used for SSL authentication.

Required Value

`com.sun.enterprise.iiop.IIOPSSLSocketFactory` (this is the configured default)

java.naming.authoritative

Description

Specifies the authoritativeness of the service requested. If **true**, the most authoritative source is to be used as specified (for example, bypass any caches, or bypass replicas in some systems). If **false**, the source need not be (but can be) authoritative.

Required Value

Enter **true** or **false**. False is the configured default.

java.naming.batchsize

Description

Specifies the preferred batch size to use when returning data using the underlined Application Server's protocol. This is a suggestion to the provider, to return the results of operations in batches of a specified size, so that the provider can optimize its performance and resources. It does not affect number or size of the data returned.

Required Value

Enter the preferred batch size. If a value is not specified, the service provider default is used.

java.naming.dns.url

Description

Specifies the DNS host and domain names (Context.DNS_URL).

Required Value

Enter a valid DNS host. If a value is not specified, the service provider default is used.

java.naming.factory.control

Description

Specifies a colon-separated list of class names of response control factory classes to be used. (LdapContext.CONTROL_FACTORIES) See **ControlFactory.getControlInstance()**.

Required Value

Enter the class names of response control factory classes, separated by a colon.

java.naming.factory.initial

Description

Specifies the class name of initial context factory. Defines the implementation of JNDI to be used by the client (Context.INITIAL_CONTEXT_FACTORY). For most cases the following configured default is used:

```
com.sun.jndi.cosnaming.CNCtxFactory
```

You can implement client-side load balancing using the Sun Java System Application Server-specific naming factory class **SIASCtxFactory** (com.sun.appserv.naming.S1ASCtxFactory). This class is used on the client-side to maintain a pool of ORB instances in order to limit the number of ORB instances that are created for a given process.

Note that the value of the name service hostname (as in **java.naming.provider.url**, or **org.omg.CORBA.ORBInitialHost**) could be a name that maps to multiple IP addresses. The **S1ASCtxFactory** will appropriately round robin ORB instances across all IP addresses every time a user calls the new **InitialContext()** method.

You can also use the following System property used by **S1ASCtxFactory** class to implement client-side load balancing if the effective name service hostname does not map to multiple IP addresses (see **com.sun.appserv.iiop.loadbalancingpolicy** on page 24). This property provides you with a list of host:port combinations to which you can round robin the ORBs. These host names may also map to multiple IP addresses. If you use this property along with **org.omg.CORBA.ORBInitialHost** and **org.omg.CORBA.ORBInitialPort** as system properties, the round robin algorithm will round robin across all the values provided.

Required Value

Enter the class name of the initial context factory to be used. The configured default is **com.sun.jndi.cosnaming.CNCtxFactory**.

java.naming.factory.object

Description

Specifies a colon-separated list of the class names of object factory classes to be used (Context.OBJECT_FACTORIES). See **NamingManager.getObjectInstance()** and **DirectoryManager.getObjectInstance()**.

Required Value

Enter the class names of object factory classes, separated by a colon.

java.naming.factory.state

Description

Specifies a colon-separated list of the class names of state factory classes to be used (Context.STATE_FACTORIES). See **NamingManager.getObjectInstance()** and **DirectoryManager.getObjectInstance()**.

Required Value

Enter the class names of state factory classes, separated by a colon.

java.naming.factory.url.pkgs

Description

Specifies a colon-separated list of package prefixes to use when loading in URL context factories. (Context.URL_PKG_PREFIXES) See **NamingManager.getURLContext()**.

Required Value

Enter the package prefixes used to load URL context factories, separated by a colon. **com.sun.jndi.url** is always added to end of list.

java.naming.language

Description

Specifies a colon-separated list of preferred languages to use with this service. Languages are specified using tags defined in **RFC 1766**. (Context.LANGUAGE)

Required Value

Language tags as specified by **RFC1776** protocol, separated by a colon. For example:

```
en-US:fr:fr-CH:ja-JP-kanji
```

If a value is not specified, the service provider default is used.

java.naming.provider.url

Description

Specifies the PROVIDER_URL (Context.PROVIDER_URL): that is, the URL of the server containing the EJB. The value set here will override any corresponding system property setting. Also, if the **java.naming.provider.url**, the **org.omg.CORBA.ORBInitialHost**, and the **org.omg.CORBA.ORBInitialPort** properties are all three set, then the value for the **java.naming.provider.url** property takes precedence.

Required Value

Enter the URL of the participating host (for example, iiop://localhost:3700).

java.naming.referral

Description

Specifies whether referrals encountered by the service provider are to be followed automatically. (Context.REFERRAL) The appropriate values are:

- **follow –** follow referrals automatically.
- **ignore –** ignore any encountered referrals.
- **throw –** throw a ReferralException when a referral is encountered.

Required Value

Enter **follow**, **ignore**, or **throw**. If a value is not specified, it uses the service provider default.

java.naming.security.authentication

Description

Specifies the security authentication scheme to use. (Context.SECURITY_AUTHENTICATION) The values are as follows:

- **simple –** provides user password authentication. Values must also be provided for java.naming.security.principal and java.naming.security.credentials parameters.
- **strong –** provides certificate authentication (a file name). May require the use of X.509 certificates for the java.naming.security.credentials property. Values must also be provided for java.naming.security.principal and java.naming.security.credentials parameters.
- **none –** no required authentication.
- ***user defined* –** a user-defined key for authentication. Values must also be provided for **java.naming.security.principal** and **java.naming.security.credentials** parameters.

Required Value

Enter a security authentication property. Values are **simple**, **strong**, **none**, or a user-defined key. If not specified it uses the service provider default.

java.naming.security.credentials

Description

Specifies the principal's (user's) credentials for the authentication scheme, determined by the authentication scheme value specified for **java.naming.security.authentication**.

If the **java.naming.security.authentication** property value is set as

- **simple** – a password is required.
- **strong** – a certificate (a file) is required.
- **none** – no value is required.
- ***user-defined*** – the user-specified key is required.

Required Value

Enter a password, certificate (file), or user-defined key depending on the value set for the **java.naming.security.authentication** property. If not specified it uses "guest", the service provider default.

java.naming.security.principal

Description

Specifies the identity of the principal (user) for the authentication scheme when the **java.naming.security.authentication** property value is set to **simple** or **strong**.

Required Value

Enter a user name or certificate, depending on the value entered for the **java.naming.security.authentication** property. If not specified, it uses "guest", the service provider default.

java.naming.security.protocol

Description

Specifies the security protocol to use (for example, "ssl").

Required Value

Enter a security protocol. If not specified it uses the service provider default.

org.omg.CORBA.ORBInitialHost

Description

This value overrides any corresponding system property setting. Also, if you set each of the following: java.naming.provider.ur, org.omg.CORBA.ORBInitialHost, and org.omg.CORBA.ORBInitialPort properties, the value for the java.naming.provider.url property will take precedence.

Required Value

Enter the host name of the org.omg.CORBA.ORBInitialHost.

org.omg.CORBA.ORBInitialPort

Description

This value overrides any corresponding system property setting. Also, if you set each of the following: java.naming.provider.ur, org.omg.CORBA.ORBInitialHost, and org.omg.CORBA.ORBInitialPort properties, the value for the java.naming.provider.url property will take precedence.

Required Value

Enter the port number for the org.omg.CORBA.ORBInitialPort.

5.4 Alerting and Logging

eGate provides an alerting and logging feature. This allows monitoring of messages, and captures any adverse messages in order of severity based on configured severity level and higher. To enable Logging, please see the ***eGate Integrator User's Guide***.

Note: *The alerts/status notifications for the Sun AppServer eWay are currently limited to Started, Running, Stopping, and Stopped.*

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

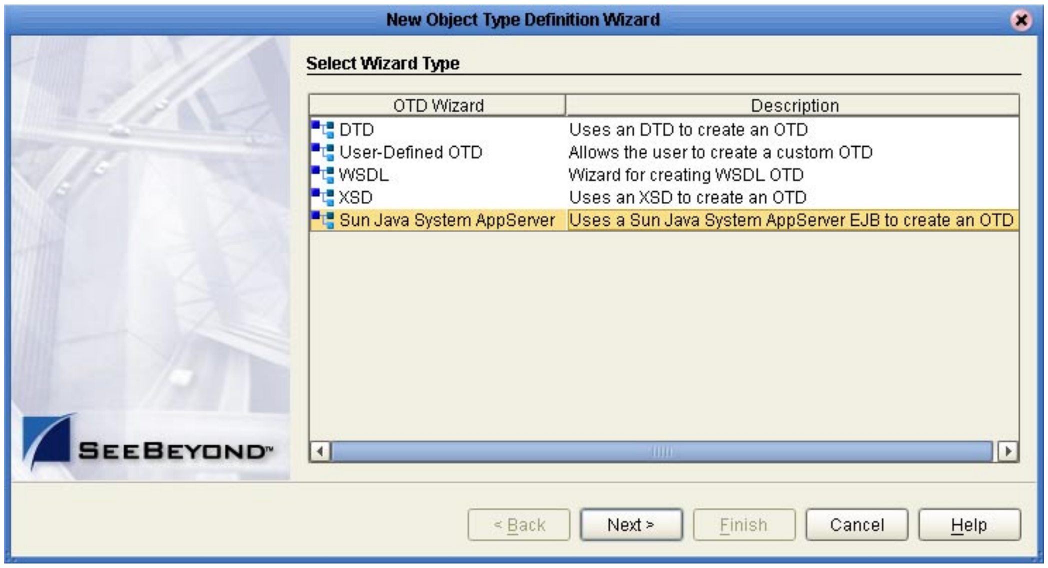
Steps required to create an OTD include:

- **Select Wizard Type** on page 30
- **Specify OTD Name** on page 31
- **Select Interfaces** on page 32
- **Select Method Argument Names** on page 33
- **Select Class Path** on page 34

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 8 on page 31**).

Figure 8 OTD Wizard Selection



OTD Wizard	Description
DTD	Uses an DTD to create an OTD
User-Defined OTD	Allows the user to create a custom OTD
WSDL	Wizard for creating WSDL OTD
XSD	Uses an XSD to create an OTD
Sun Java System AppServer	Uses a Sun Java System AppServer EJB to create an OTD

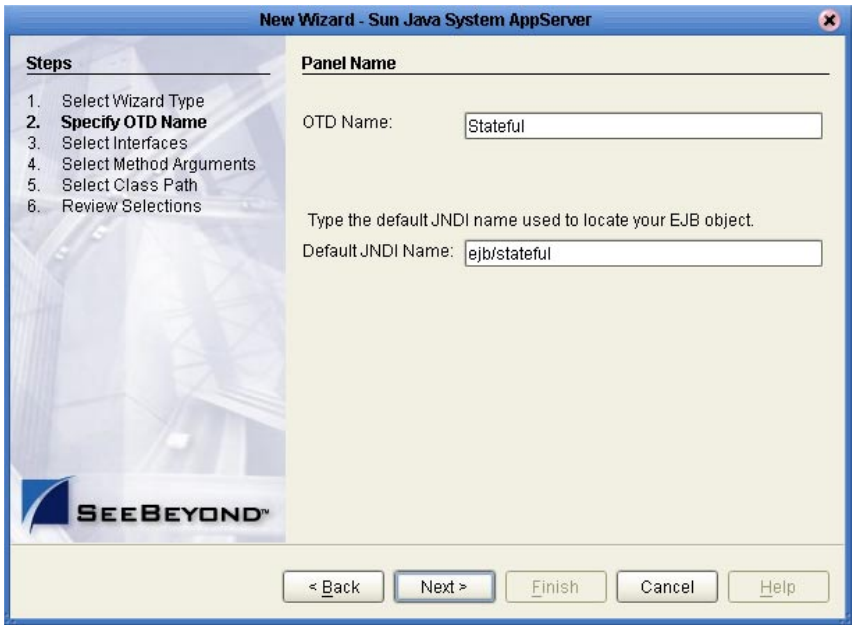
3 From the list, select the **Sun Java System AppServer** OTD Wizard and click **Next**. The **Specify OTD Name** page appears.

Specify OTD Name

4 Enter a name for the new OTD.

5 Enter the Java Naming and Directory Interface (JNDI) name used to locate your Enterprise Java Bean (EJB) object (see Figure 9).

Figure 9 Database Connection Information



OTD Name: Stateful

Type the default JNDI name used to locate your EJB object.

Default JNDI Name: ejb/stateful

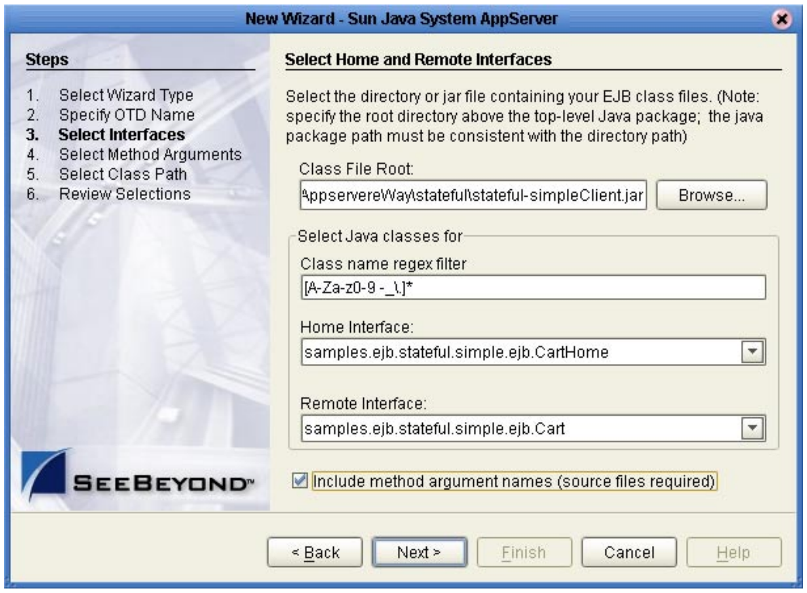
6 Click **Next**, the Select Interfaces page appears.

Select Interfaces

From the **Select Interfaces** page, locate and select the root directory or specific JAR file containing the EJB class files. Available Fields Include:

- **Class File Root** – contains the path of the EJB (archive) JAR file or a directory path that includes the EJB, and other dependent JAR files.
- **Class name regex** – used to enter regular expressions that describe a pattern to match in text.
- **Home Interface** – used to define the enterprise bean's life-cycle methods that are used to create, remove, and find beans.
- **Remote Interface** – used to defines an enterprise bean's business methods, which are used by bean clients to interact with the bean.
- **Include method argument names** option - used to add the method argument names to the OTD. Do not select this option if you do not have the EJB source code.

Figure 10 Select Interfaces page



Select Home and Remote Interfaces

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)

Class File Root: AppservereWay\stateful\stateful-simpleClient.jar

Select Java classes for

Class name regex filter: [A-Za-z0-9 -_\.]*

Home Interface: samples.ejb.stateful.simple.ejb.CartHome

Remote Interface: samples.ejb.stateful.simple.ejb.Cart

✓ Include method argument names (source files required)

7 Click the **Browse** button next to the Class File Root field, and navigate to the required EJB JAR file. The EJB JAR file must contain both the Home and Remote Interface classes for the EJB.

The stub class files for the Home and Remote interfaces must be included, otherwise runtime problems may occur.

If you provide an EJB class file that does not contain all necessary dependent class files required for the Home or Remote Interface, a **Confirm Selection** prompt appears, allowing you to provide additional dependent class files. The **Confirm Selection** dialog box includes the following options:

- **Yes** – provides the additional classpath and name of required JAR file.
- **No** – ignores a class that has a dependency.
- **Cancel** – ignores all classes with dependencies.

If you do not specify a required 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.

8 Select the JAR file and click **Open**. Both the Home Interface and Remote Interface fields are automatically populated (see **Figure 10 on page 32**).

9 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.

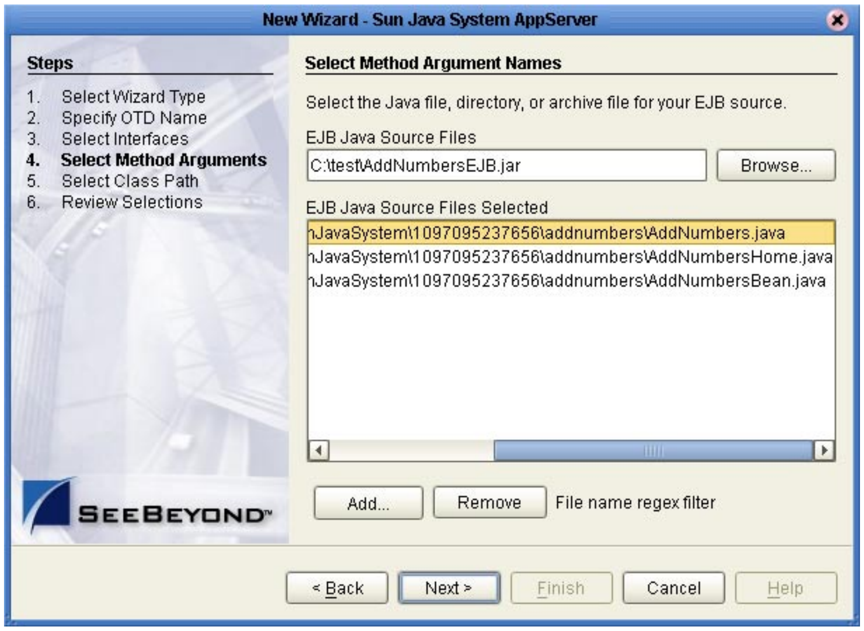
10 Click **Next**. Do one of the following:

- If you selected the **Include method argument names** option, the **Select Method Argument Names** page (wizard step 5) appears. Proceed to **Select Method Argument Names** on page 33.
- If the selected EJB file's interfaces are valid and the Select Method argument names option was not selected, the wizard advances to the **Select Class Path** page. Proceed to **Select Class Path** on page 34.

Select Method Argument Names

11 The **Select Method Argument Names** page appears. Use **Browse** to locate the necessary EJB Java source files.

Figure 11 Select Method Argument page



Select the Java file, directory, or archive file for your EJB source.

EJB Java Source Files: C:\test\AddNumbersEJB.jar

EJB Java Source Files Selected:

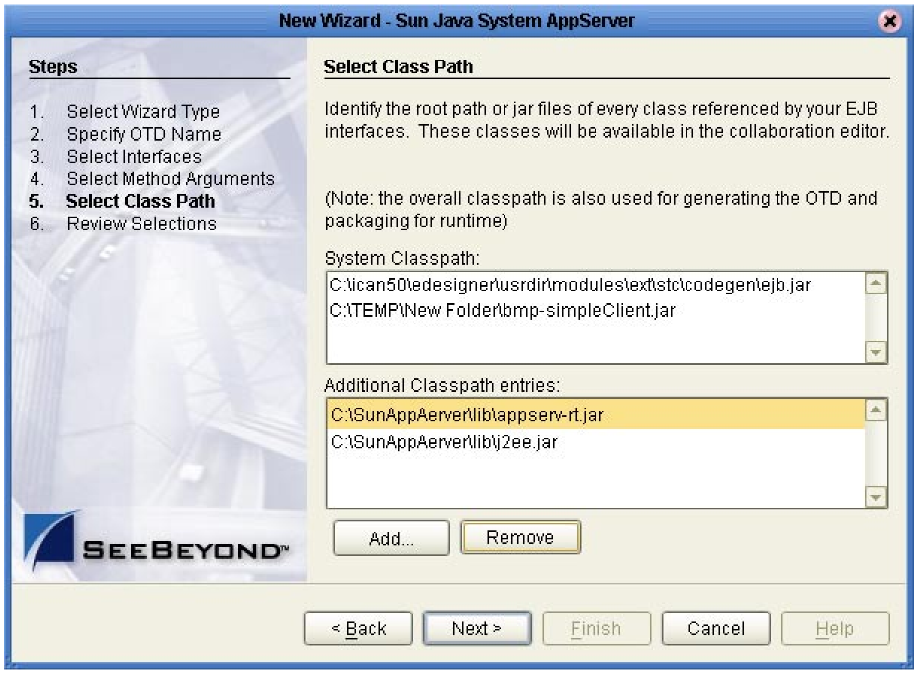
```
nJavaSystem\1097095237656\addnumbers\AddNumbers.java
nJavaSystem\1097095237656\addnumbers\AddNumbersHome.java
nJavaSystem\1097095237656\addnumbers\AddNumbersBean.java
```

12 Use the **Add** option to locate and add Java source files to the **EJB Java Source Files Selected** field. Remove unwanted files from the **EJB Java Source Files Selected** field by selecting the file and clicking **Remove**. Once the field contains all the necessary files, click **Next**.

Select Class Path

13 Click the **Next** button. The Select Class Path window appears, containing a list of system based JAR files referenced by the EJB. You must identify the root path and JAR files of every class referenced by the EJB interfaces (see Figure 12).

Figure 12 Select Class Path page



Select Class Path

Identify the root path or jar files of every class referenced by your EJB interfaces. These classes will be available in the collaboration editor.

(Note: the overall classpath is also used for generating the OTD and packaging for runtime)

System Classpath:

C:\ican50\edesigner\usrdir\modules\ext\stc\codegen\ejb.jar
C:\TEMP\New Folder\bmp-simpleClient.jar

Additional Classpath entries:

C:\SunAppAerver\lib\appserv-rt.jar
C:\SunAppAerver\lib\j2ee.jar

The **Additional Classpath Entries** field must include the following files:

A **C:\SunAppServer\lib\j2ee.jar**

B **C:\SunOneAppServer\lib\appserv-rt.jar**, or **C:\SunOneAppServer\lib\appserv-ext.jar**

Appserv-rt.jar is a larger file which includes configuration for Client Side Load Balancing. If Client Side Load Balancing will **NOT** be configured, the smaller **appserv-ext.jar** file can be used. For more information, see **java.naming.factory.initial** on page 25 and also refer to Sun's documentation regarding RMI/IIOP EJB Client.

Also includes any other files (application specific to the EJB at hand) containing any classes that the EJB files are dependent upon. The wizard will analyze the dependency and prompt the user for the location(s) of the files in the first panel.

14 Click the **Add** button. From the **Open** window, navigate to and select the required JAR files. Click **Open** to add the new classpath entry to the Additional Class Path field. Repeat until all the required classpaths entries are listed. Click **Next**.

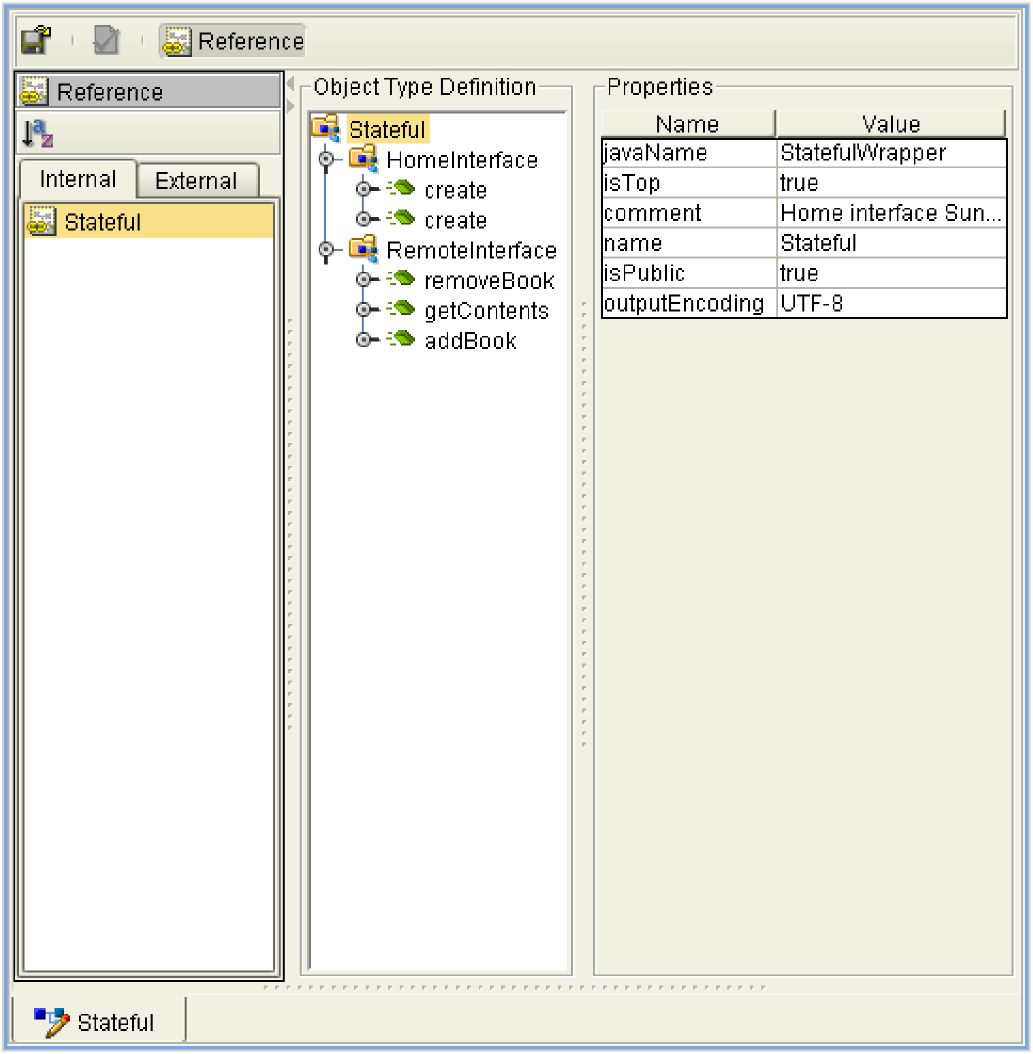
Review Selections

15 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

16 Once you are satisfied with your selection information, click **Finish**. The OTD Editor appears with the generated OTD (see Figure 13).

Figure 13 OTD Editor - New OTD



Name	Value
javaName	StatefulWrapper
isTop	true
comment	Home interface Sun...
name	Stateful
isPublic	true
outputEncoding	UTF-8

Chapter 7

Implementing a Project Using eInsight

This chapter describes how to use the Sun AppServer eWay with the ICAN 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 *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

This following pages provide directions for creating a simple project, **SunAppSvr_BP_Sample**, that demonstrates how eInsight Business Processes are used with the Sun AppServer eWay. The same project can be uploaded from the Installation CD-ROM in a near-complete state.

Sample Overview

The Sun AppServer eWay project, **SunAppSvr_BP_Sample**, 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 SunAppSvr_BP 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 ICAN projects see the *eInsight Business Process Manager User's Guide* and the *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 *eInsight Business Process Manager User's Guide*.

7.5 Importing a Sample Project

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 Enterprise Manager's Documentation tab. Extract the Sun Java System Application Server eWay samples from the Enterprise Manager to a local file.
2. Save all your unsaved work before importing a sample 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 example, **SunAppSvr_BP_Sample.zip**) and click **Import**. After the sample project is successfully imported, click **Close**.

7.6 Running a Sample Project

Before an imported sample project can be run you must do the following:

- Create an **Environment** (see **Creating an Environment** on page 49)
- Configure the eWays for your system (see **Configuring the eWays** on page 50)
- Create a **Deployment Profile** (see **Creating and Activating the Deployment Profile** on page 52)

7.7 Creating the SunAppSvr_BP_Sample Project

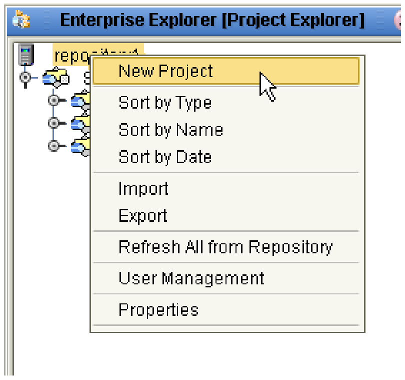
The following pages provide step by step directions for manually creating the SunAppSvr_BP_Sample project.

7.7.1. 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 14 on page 39**). A new project (**Project1**) appears on the Project Explorer tree.

Figure 14 Enterprise Explorer - New Project



2 Click twice on **Project1** and rename the project (for this sample, **SunAppSvr_BP_Sample**).

7.7.2 Creating a Connectivity Map

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

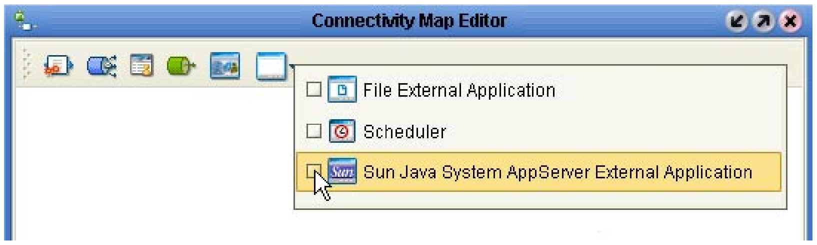
1 From the Project Explorer tree, right-click the new **SunAppSvr_BP_Sample** project and select **New > Connectivity Map** from the shortcut menu.

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**.

Selecting the External Applications

In the Connectivity Map, the eWays are associated with External Systems. 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 15).

Figure 15 Connectivity Map - External Applications


Connectivity Map Editor
File External Application
Scheduler
Sun Java System AppServer External Application

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

2 Select the external systems necessary to create your project (for this sample, **Sun Java System AppServer External Application** and **File**). Icons representing the selected external systems 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.

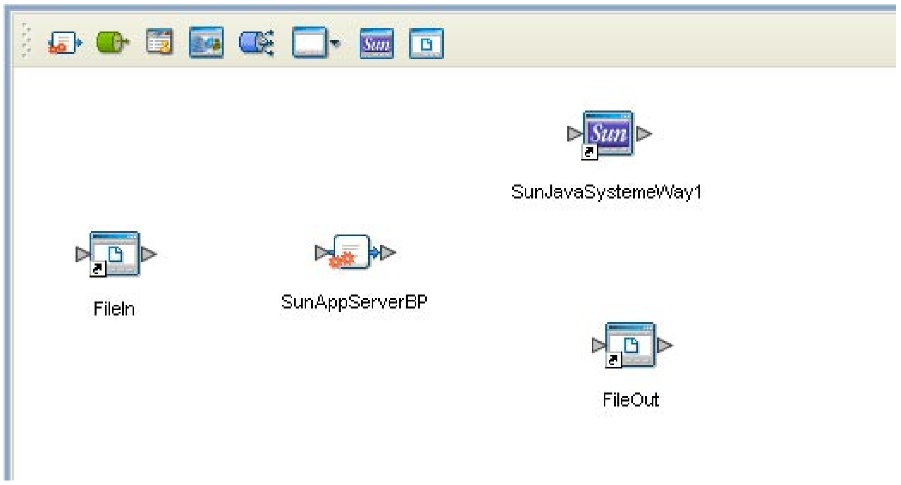
1 For this sample, drag the following components onto the Connectivity Map canvas as displayed in Figure 16:

- **File External System** (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 System** (1 for this sample)

2 Rename the components on the Connectivity Map as follows:

- **File1** External Application to **FileIn**
- **File2** External Application to **FileOut**
- Rename the **Service1** Service to **SunAppServerBP**

Figure 16 Connectivity Map with Components



3 Save your current changes to the Repository.

7.7.3. 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 files (for example, bmp-simpleClient.jar), provided with the Sun Java System Application Server Samples Bundle (for example, bmp-simpleClient.jar), downloadable from **http://java.sun.com**. Make sure that this sample bundle is installed prior to manually creating the sample.

1 From the Project Explorer tree, right-click the **SunAppServer_JCE_Sample** project and select **New** > **Object Type Definition** from the shortcut menu. The **Object Type Definition Wizard** appears.

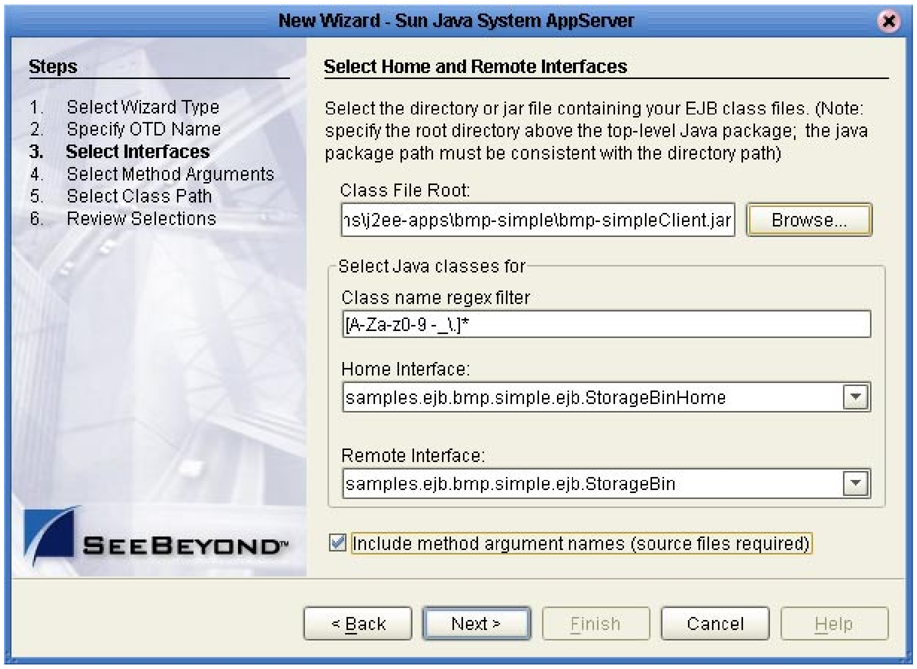
2 From the Select Wizard Type box, select **Sun Java System AppServer Wizard** and click **Next**.

3 For **step 2** of the wizard (**Specify OTD Name**), enter **Storage** as the OTD Name, enter **ejb/MyStorageBin** as the Default JNDI Name, and click Next.

4 For **step 3** of the wizard (**Select Interfaces**), browse to the location of the Sun Java System Application Server sample JAR file, **bmp-simpleClient.jar**. Click **Select** to add the file to the Selected Files box.

5 Select **Include method argument names** (see Figure 17), and click **Next**.

Figure 17 Sun Java System AppServer OTD Wizard - Step 3


New Wizard - Sun Java System AppServer

Steps
1. Select Wizard Type
2. Specify OTD Name
3. **Select Interfaces**
4. Select Method Arguments
5. Select Class Path
6. Review Selections

Select Home and Remote Interfaces

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)

Class File Root: ns\j2ee-apps\bmp-simple\bmp-simpleClient.jar [Browse...]

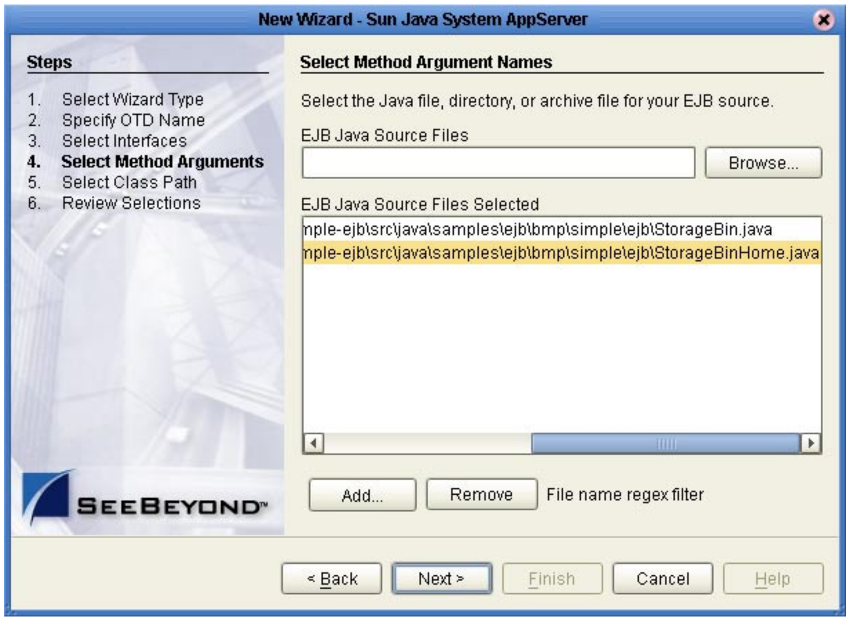
Select Java classes for
Class name regex filter: [A-Za-z0-9 -_\.]*
Home Interface: samples.ejb.bmp.simple.ejb.StorageBinHome
Remote Interface: samples.ejb.bmp.simple.ejb.StorageBin

☑ Include method argument names (source files required)

6 The wizard proceeds to step 5, **Select Method Arguments**. From the EJB Java Source Files field, browse to and select **StorageBin.java**. Locations vary depending on the sample version and system (example **C:\Sun\AppServer\samples\ejb\bmp\apps\simple\simple-ejb\src\java\samples\ejb\bmp\simple\ejb)**. The file is added to the **EJB Java Source Files Selected** field.

7 Click **Add**, and from the **Open** page, select **StorageBinHome.java**. Click **Open**. The file is added to the **EJB Java Source Files Selected** field (see **Figure 18 on page 42**).

Figure 18 Sun Java System AppServer OTD Wizard - Step 5


New Wizard - Sun Java System AppServer

Steps
1. Select Wizard Type
2. Specify OTD Name
3. Select Interfaces
4. **Select Method Arguments**
5. Select Class Path
6. Review Selections

Select Method Argument Names

Select the Java file, directory, or archive file for your EJB source.

EJB Java Source Files

EJB Java Source Files Selected

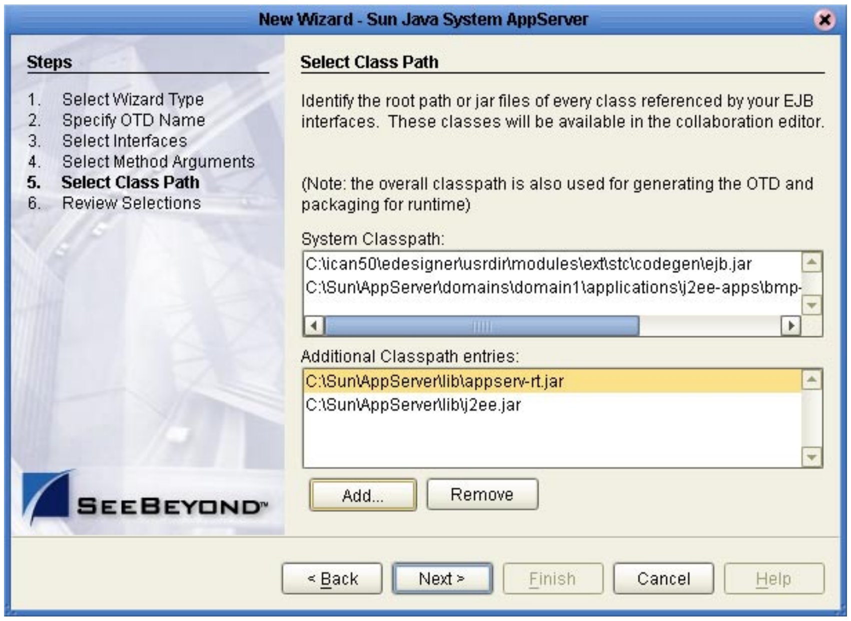
npIe-ejb\src\java\samples\ejb\bmp\simple\ejb\StorageBin.java
npIe-ejb\src\java\samples\ejb\bmp\simple\ejb\StorageBinHome.java

File name regex filter

8 Click Next, The wizard proceeds to step 6, **Select Class Path**. Click **Add**, locate and select **j2ee.jar**. The file is added to the **Additional Classpath entries** field.

9 Click the Add button again, locate and select **appserv-rt.jar** to add the file to the **Additional Classpath entries** field (see Figure 19).

Figure 19 Sun Java System AppServer OTD Wizard - Step 6


New Wizard - Sun Java System AppServer

Steps
1. Select Wizard Type
2. Specify OTD Name
3. Select Interfaces
4. Select Method Arguments
5. **Select Class Path**
6. Review Selections

Select Class Path

Identify the root path or jar files of every class referenced by your EJB interfaces. These classes will be available in the collaboration editor.

(Note: the overall classpath is also used for generating the OTD and packaging for runtime)

System Classpath:

C:\ican50\edesigner\usrdir\modules\ext\stc\codegen\ejb.jar
C:\Sun\AppServer\domains\domain1\applications\j2ee-apps\bmp-

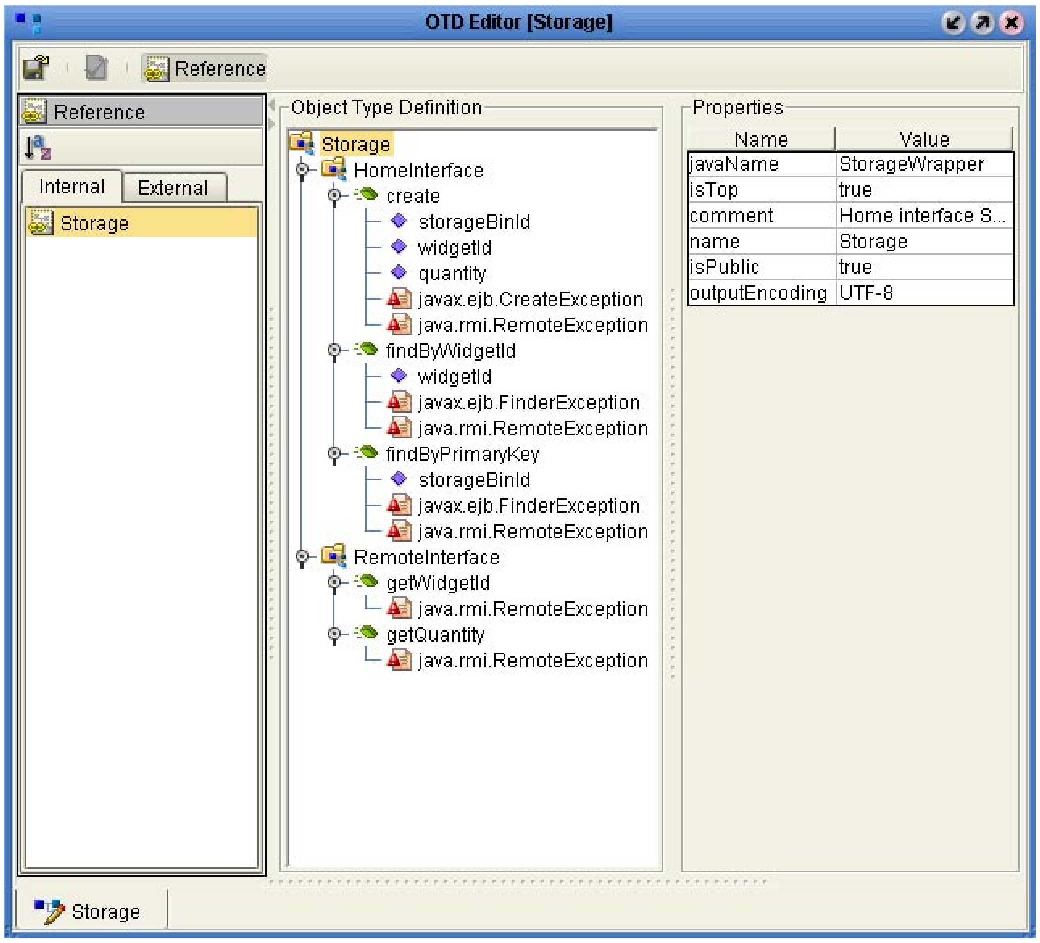
Additional Classpath entries:

C:\Sun\AppServer\lib\appserv-rt.jar
C:\Sun\AppServer\lib\j2ee.jar

10 Click **Next**. A warning appears stating that the class "com.sun.appserv.naming.S1ASCtxFactory" is not detected on the current classpath. This warning can be dismissed if load balancing will not be configured. Proceed past these warnings to the **Review Selections** step.

11 Review your selected information. When you are satisfied with your selections, click **Finish**. The OTD Editor appears with the generated **Storage** OTD (see Figure 20).

Figure 20 OTD Editor - Storage OTD



Name	Value
javaName	StorageWrapper
isTop	true
comment	Home interface S...
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 30

7.7.4 Creating the Business Process

The SunAppSvr_BP_Sample contains the Business Process, SunAppSvr_BP. To create this Business Process, do the following:

1 Right-click the new project in the Enterprise Designer's 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 **SunAppSvr_BP**.

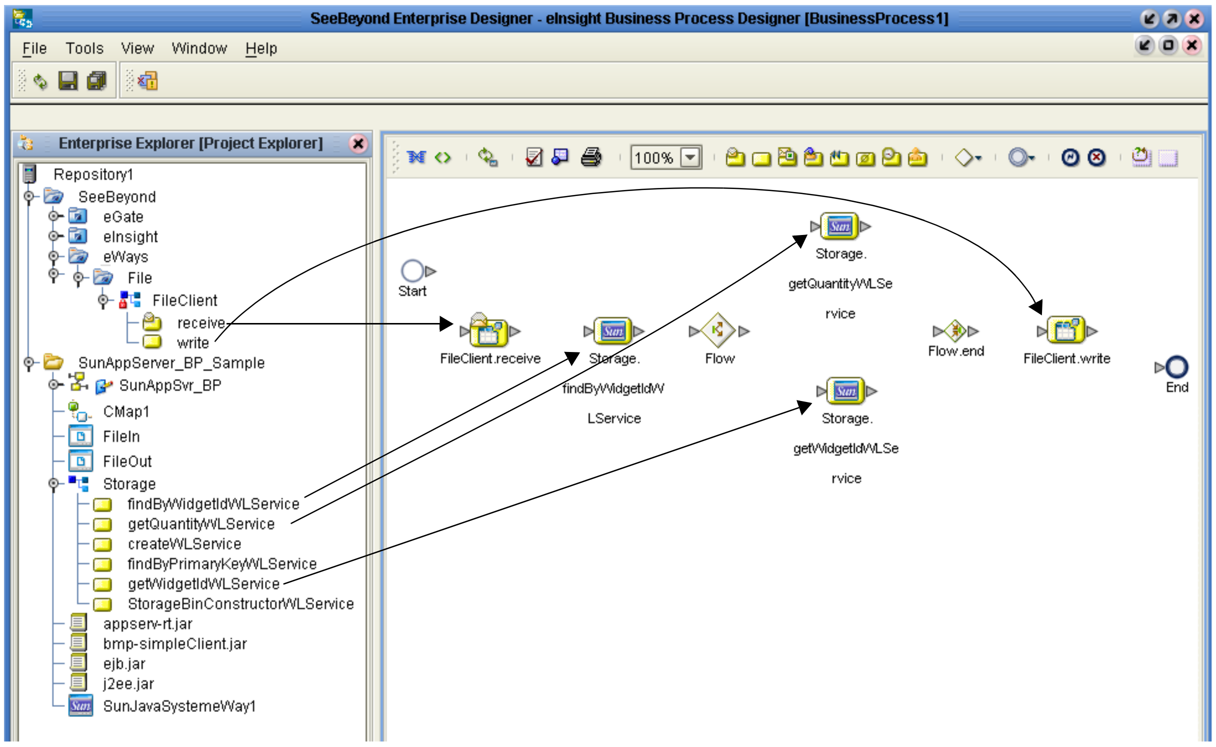
2 From the Project Explorer tree, expand **SeeBeyond > eWays > File > FileClient**, and the **Storage** OTD's nodes.

3 Populate the eInsight Business Process Designer canvas with the following activities from the Project Explorer tree, as displayed in Figure 21.

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

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 21).

Figure 21 eInsight Business Process Designer - Populate the Canvas

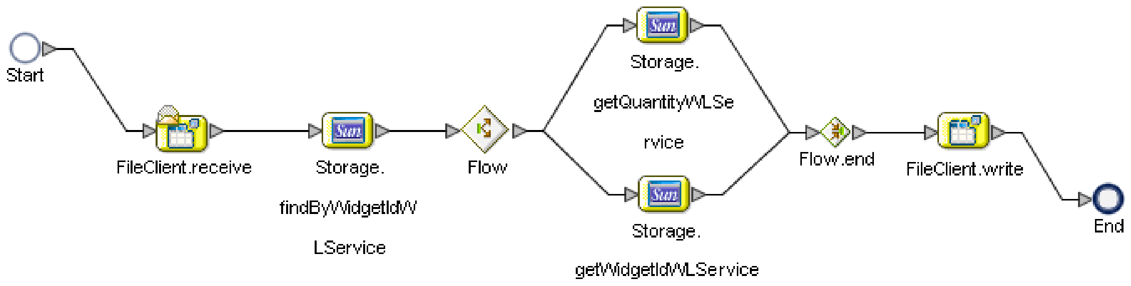


5 Link the elements by clicking on the element connector and dragging the cursor to the next element connector, making the following links as displayed in Figure 22 on page 45.

- **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 22 Business Process Designer - Link the Modeling Elements



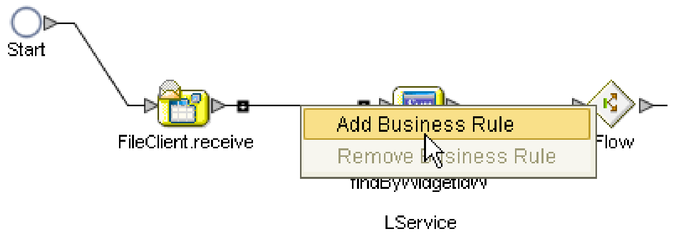
Configuring the SunAppSvr_BP Modeling Elements

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 **SunAppSvr_BP** 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 23). A Business Rule Icon is added to the link.

Figure 23 eInsight Business Process Designer - Adding Business Rules



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

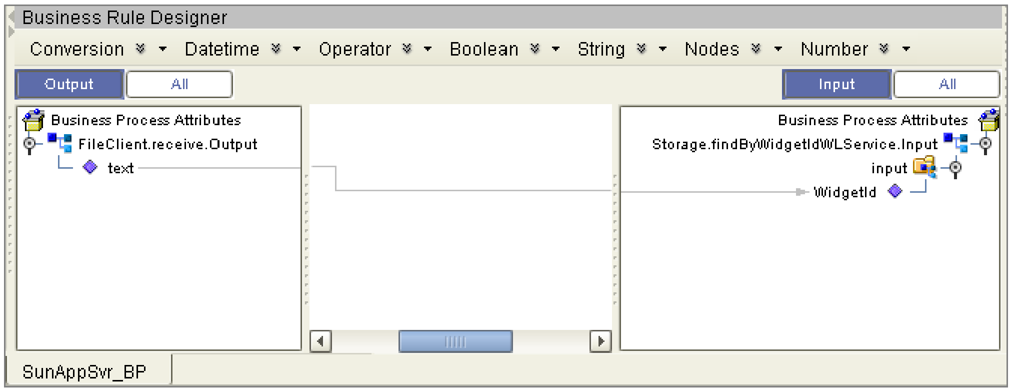
   A. Double-click the Business Rule icon. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.

   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. These Attributes can now be modified.

C Map **text**, under **FileClient.receive.Output** in the Output pane of the Business Rule Designer, to **WidgetId** 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 **WidgetId** in the Input pane. A line now connects the two nodes in the Business Rule Designer (see Figure 24).

Figure 24 FileClient.receive -> Storage.findByWidgetIdWLService Rule



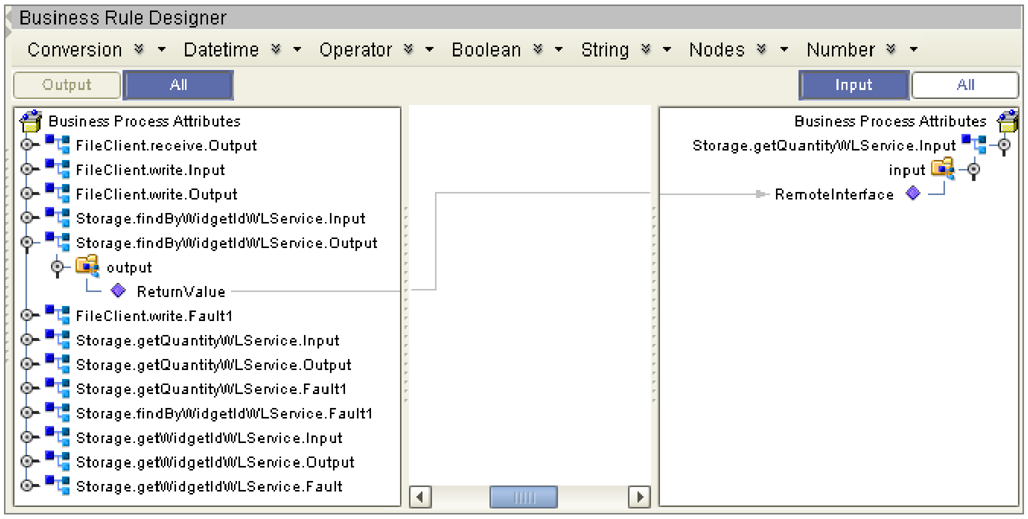
3 Create the **Flow -> Storage.getQuantityWLService** Business Rule by doing the following:

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

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

C Map **ReturnValue**, under **Storage.findByWidgetIdWLService.Output -> 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 24).

Figure 25 Flow -> Storage.getQuantityWLService Rule


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

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.Output -> output** in the Output pane of the Business Rule Designer, to **RemoteInterface** under **Storage.getWidgetIdWLService.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. Repeat this step twice more to create three **concat** method boxes.

D From the Business Rule Designer's String menu, select **string literal**. The Input dialog box appears. Enter **WidgetID:** as the value and click **OK**. The new **string literal** method box appears.

E From the Business Rule Designer's String menu, select **string literal**. The Input dialog box appears. Enter   **Quantity** as the value (leaving 7 spaces before the word) and click **OK**. The new **string literal** method box appears.

F Map the **WidgetID** output node of the first string literal method box, to the **string1** input node of the first concat method box.

G Map the   **Quantity** output node of the second string literal method box, to the **string1** input node of the second concat method box.

H Map **ReturnValue**, under **Storage.getWidgetIdWLService.Output -> output** in the Output pane of the Business Rule Designer, to the **string2** input node of the first concat method box.

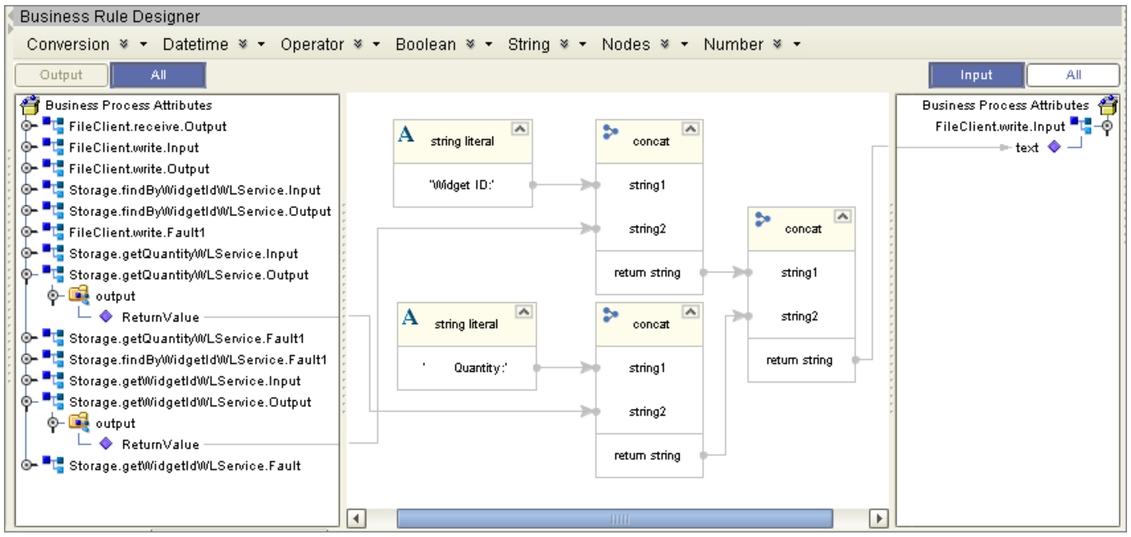
I Map **ReturnValue**, under **Storage.getQuantityWLService.Output -> output** in the Output pane of the Business Rule Designer, to the **string2** input node of the second concat method box.

J Map the **return string** output node of the first concat method box, to the **string1** input node of the third concat method box.

K Map the **return string** output node of the second concat method box, to the **string2** input node of the third concat method box.

L Map the **return string** output node of the third concat method box, to **text**, under **FileClient.write.Input** in the Input pane of the Business Rule Designer (see **Figure 26 on page 48**).

Figure 26 Flow.end -> FileClient.write Rule


Business Rule Designer
Conversion
Datetime
Operator
Boolean
String
Nodes
Number
Output
All
Input
All
Business Process Attributes
FileClient.receive.Output
FileClient.write.Input
FileClient.write.Output
Storage.findByWidgetIdWLService.Input
Storage.findByWidgetIdWLService.Output
FileClient.write.Fault1
Storage.getQuantityWLService.Input
Storage.getQuantityWLService.Output
output
ReturnValue
Storage.getQuantityWLService.Fault1
Storage.findByWidgetIdWLService.Fault1
Storage.getWidgetIdWLService.Input
Storage.getWidgetIdWLService.Output
output
ReturnValue
Storage.getWidgetIdWLService.Fault
string literal
'Widget ID:'
concat
string1
string2
return string
string literal
' Quantity:'
concat
string1
string2
return string
concat
string1
string2
return string
Business Process Attributes
FileClient.write.Input
text

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.

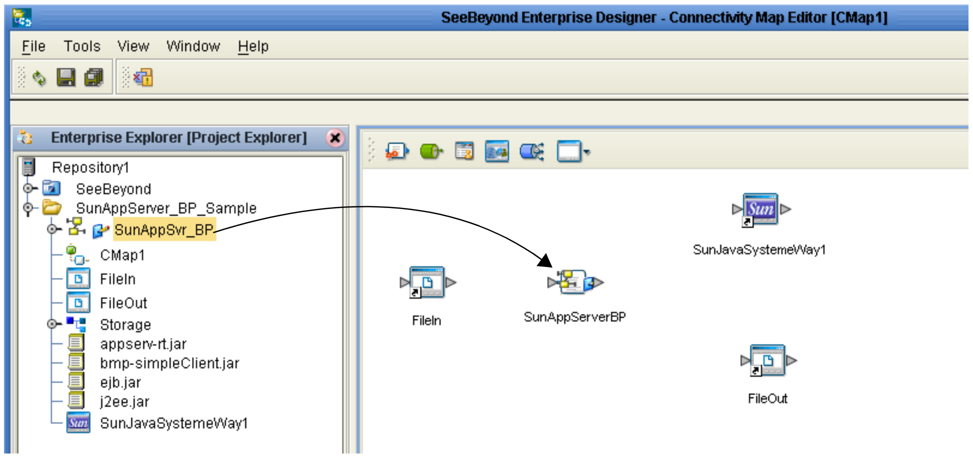
7.7.5. Binding the eWay Components

After the Business Processes have been completed, the components are associated and Bindings are created in the Connectivity Map.

1 From the Project Explorer, double-click **CMap1** to display the Connectivity Map.

2 Drag and drop the **SunAppSvr_BP** Business Process from the Project Explorer tree to the **SunAppSvrBP** Service. If the Business Process was successfully associated, the Service's icon changes to a Business Process icon (see Figure 27).

Figure 27 Connectivity Map - Associating the Project's Components



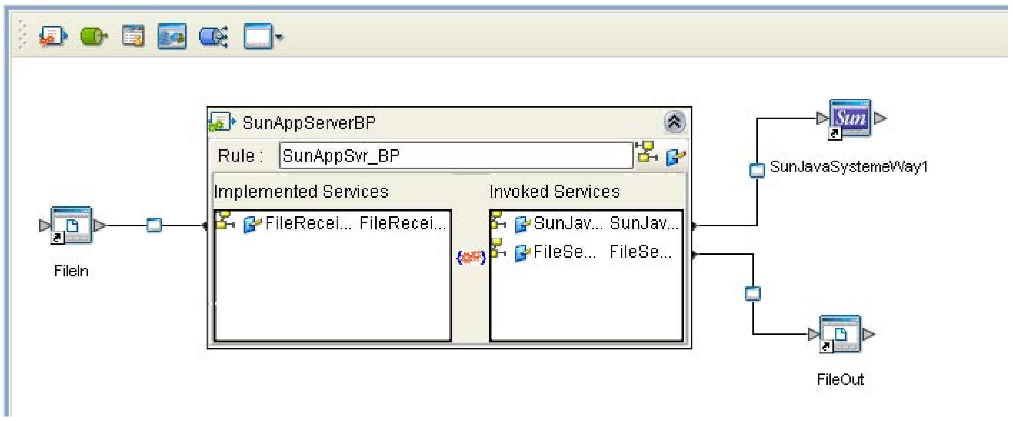
3 Double-click **SunAppSvrBP**. The **SunAppSvrBP** binding dialog box appears using the **SunAppSvr_BP** Rule.

4 From the **SunAppSvrBP** binding dialog box, drag **FileReceiver** (under Implemented Services) to the **FileIn** (File) External Application.

5 From the **SunAppSvrBP** binding dialog box, drag **SunFileSystem_Storage** (under Invoked Services) to the **SunJavaSystemeWay1** External Application.

6 From the **SunAppSvrBP** binding dialog box, drag **FileSender** (under Invoked Services) to the **FileOut** External Application (see Figure 28).

Figure 28 Connectivity Map - Binding the Project's Components



7 Save your current changes to the Repository.

7.7.6. 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 Enterprise Designer's Environment Explorer and Environment Editor.

1 From the Enterprise Designer's 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 **SunAppSvr_BP_Sample_ENV**.

4 Right-click **SunAppSvr_BP_Sample_ENV** and select **New Sun Java System AppServer External System**. Name the External System **SunAppSvrExtSys.** Click **OK**. **SunAppSvrExtSys** is added to the Environment Editor.

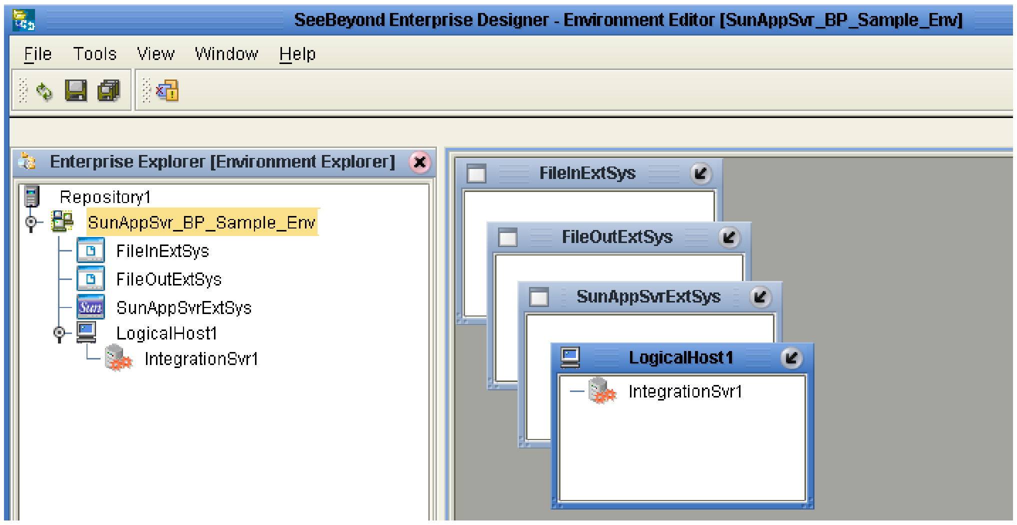
5 Right-click **SunAppSvr_BP_Sample_ENV** and select **New File External System**. Name the External System **FileExtSysIn** and select **Inbound File eWay** as the External System Type. Click **OK**. **FileExtSysIn** is added to the Environment Editor.

6 Right-click **SunAppSvr_BP_Sample_ENV** and select **New File External System**. Name this External System **FileExtSysOut** and select **Outbound File eWay** as the External System Type. **FileExtSysOut** is added to the Environment Editor.

7 Right-click **SunAppSvr_BP_Sample_ENV** and select **New Logical Host**. **LogicalHost1** is added to the Environment Editor.

8 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost1 (seeFigure 29).

Figure 29 Environment Editor



9 Save changes to the repository.

7.7.7. Configuring the eWays

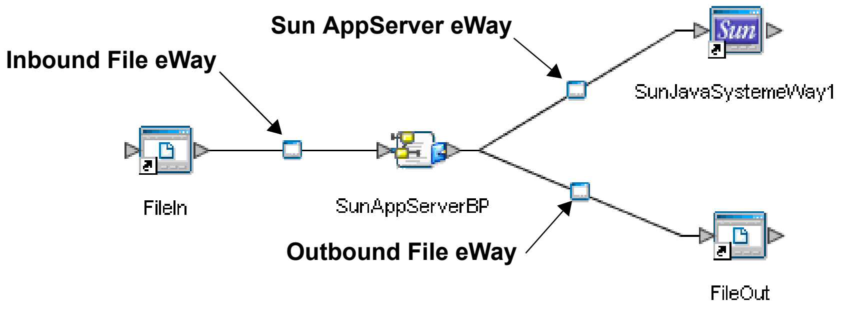
The SunAppServer_BP_Sample project uses three eWays, each represented in the Connectivity Map as a node between an External Application and a Service. eWays facilitate communication and movement of data between the external applications and the eGate system.

The File eWay properties are configured from the Connectivity Map. The Sun AppServer eWay properties are set from both the Project Explorer's Connectivity Map and the Environment Explorer tree.

Configuring the File eWay Properties

1 Double-click the inbound **FileIn eWay** (see Figure 30), select **Inbound File eWay** in the Templates dialog box and click **OK**.

Figure 30 eWays



2 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 2, and click **OK**.

Table 2 Inbound File eWay Settings

Inbound eWay Connection Parameters	
Directory	C:/temp
Input file name	Input*.txt
Polling interval	5000

3 In the same way, modify the outbound File eWay properties for your system, including the settings in Table 3, and click **OK**.

Table 3 Outbound File eWay Settings

Outbound eWay Connection Parameters	
Directory	C:/temp
Output file name	output%d.dat

Configuring the Sun AppServer eWay Properties

The Sun AppServer eWay properties are set in both the Connectivity Map and the Environment Explorer. In most cases, the default configuration is sufficient and no change is necessary. For more information on the Sun AppServer eWay properties and the Properties Editor, see **Configuring the Sun AppServer eWay Properties** on page 19 or see the *eGate Integrator User's Guide*.

Modifying 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 4, and click **OK**.

Table 4 Sun AppServer eWay Connectivity Map Properties

Inbound Sun AppServer eWay (Connectivity Map) Properties	
JNDI InitialContext Settings Set as directed, otherwise use the default settings	
JNDI name	ejb/MyStorageBin *(for example)*

Modifying the Sun AppServer eWay Environment Explorer Properties

1 From the **Environment Explorer** tree, right-click the Sun AppServer eWay External System (**SunAppSvrExtSys** 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. For this sample, the default settings should be adequate.

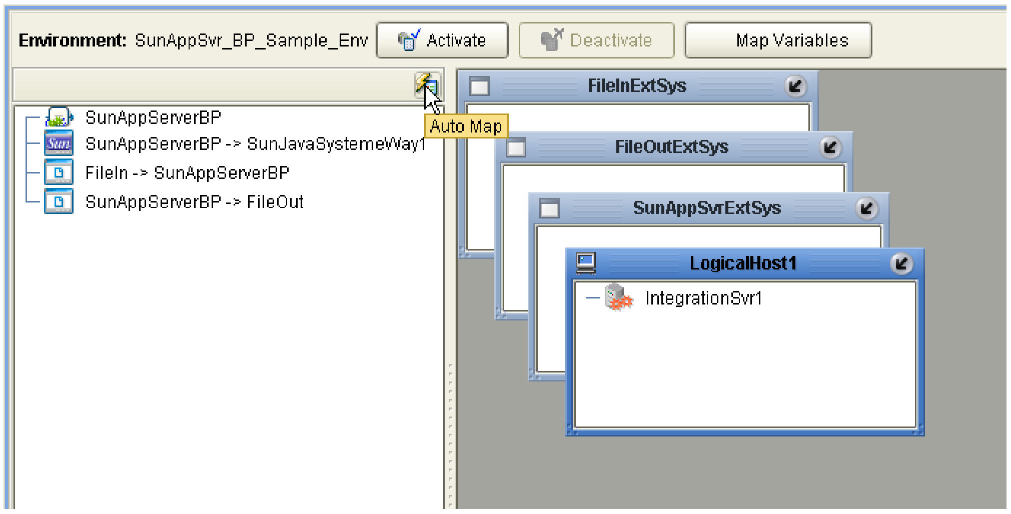
7.7.8 Creating and Activating the Deployment Profile

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 (using the SunAppSvr_BP_Sample), do the following:

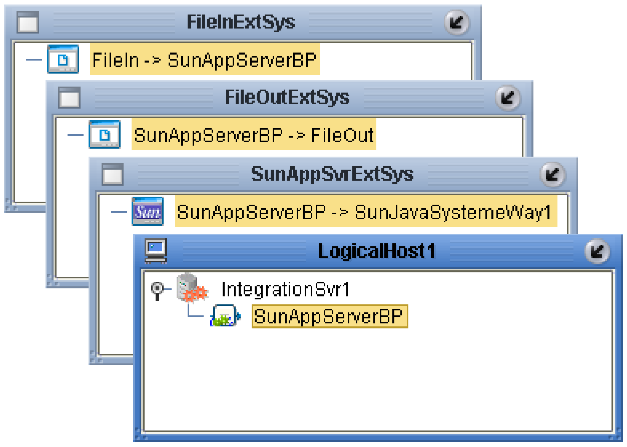
1. From the Enterprise Explorer's Project Explorer, right-click the project (for example, **SunAppServer_BP_Sample**) and select **New** > **Deployment Profile**.
2. From the **Create Deployment Profile** dialog box, enter a name for the Deployment Profile (for example, **SunAppServer_BP_Sample_DP**), select the appropriate Environment, and click **OK**.
3. Click the **Auto Map** icon as displayed in Figure 31. The projects components are automatically mapped to their system window as seen in Figure 32 on page 53. If any of the project components are not mapped automatically after Auto Map is used, those component can be mapped manually by following the appropriate steps below. Once all components are mapped, proceed to step 8.

Figure 31 Deployment Profile - Auto Map


Environment: SunAppSvr_BP_Sample_Env

4. Drag **SunAppServerBP** (service) to **IntegrationSvr1** in the **LogicalHost1** window.
5. From the left pane of the Deployment Editor, drag **FileIn -> SunAppServerBP** to the **FileInExtSys** window.
6. From the left pane of the Deployment Editor, drag **SunAppServerBP -> SunJavaSystemeWay1** to the **SunAppSvrExtSys** window.
7. From the left pane of the Deployment Editor, drag **SunAppServerBP -> FileOut** to the **FileOutExtSys** window (see Figure 32 on page 53).

Figure 32 Deployment Profile



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

7.7.9. Running the Project

The following directions assume that the Enterprise Designer was downloaded to C:\ican50. If this is not the case, replace that location in the following directions with the appropriate location.

1 From the Enterprise Manager Downloads tab, download **Logical Host**.

2 Extract the file to the **ican50\LogicalHost1** directory. You must specify the **LogicalHost1** directory for it to be created.

3 Navigate to **C:\ican50\LogicalHost1\logicalhost\bootstrap\config** directory and open the **logical-host.properties** file using a text editor.

4 Enter the following information in the appropriate fields:

- Logical Host root directory: **ican50\LogicalHost1\logicalhost**
- Repository URL: **http://localhost:*port number*/*repository name***
- Repository user name and password: ***Your user name and password***
- Logical Host Environment name: **SunAppSvr_BP_Sample_Env**
- Logical Host name: **LogicalHost1**

Save your changes to **logical-host.properties** and close the file.

5 Run the **bootstrap.bat** file in the **ican50\LogicalHost1\logicalhost\bootstrap\bin** directory.

6 Copy the sample input data file to the input directory.

Chapter 8

Implementing a Project Using the Collaboration Editor (Java)

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 SeeBeyond Enterprise Designer. For more information on creating an eGate project see the *eGate Tutorial* and the *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 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 AppServer eWay Properties** on page 19.

Sun Java System AppServer OTD Wizard

The **Sun Java System AppServer OTD Wizard** builds an Object Type Definition from an EJB or JAR file. The wizard generates Object Type Definitions (OTDs) 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 **SunAppSvr_JCE_Sample** project that demonstrates how the Sun AppServer eWay is used with Java Collaborations. The same project can be uploaded from the Installation CD-ROM in a near-complete state.

Sample Overview

The **SunAppServer_JCE_Sample** projectdemonstrates 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 SunAppSvr_Collab 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

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 Enterprise Manager's Documentation tab. Extract the samples from the Enterprise Manager to a local file.
2. Save all unsaved work before importing a sample 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 example, **SunAppSrvr_JCE_Sample.zip**) and click **Import**. After the sample project is successfully imported, click **Close**.

8.4 Running a Sample Project

Before an imported sample project can be run you must do the following:

- Create an **Environment** (see **Creating an Environment** on page 69)
- Configure the eWays for your system (see **Configuring the eWays** on page 70)
- Create a **Deployment Profile** (see **Creating and Activating the Deployment Profile** on page 71)

8.5 Creating the SunAppServer_JCE_Sample Project

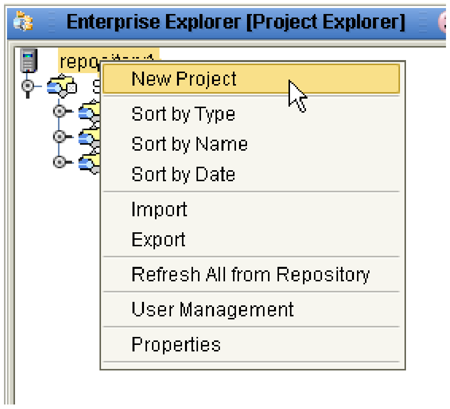
The following pages provide step by step directions for manually creating the sample project's components.

8.5.1. 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 33). A new project (**Project1**) appears on the Project Explorer tree.

Figure 33 Enterprise Explorer - New Project



2 Rename the project to **SunAppServer_JCE_Sample**.

8.5.2 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 **SunAppServer_JCE_Sample** 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**.

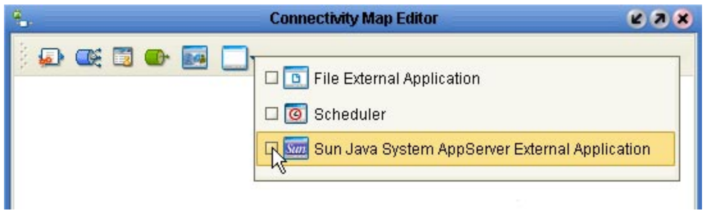
The icons in the toolbar represent the available components used to populate the Connectivity Map canvas.

Selecting the External Applications

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas.

In a Connectivity Map, the eWays are associated with External Systems. For example, to establish a connection to Sun Application Server, you must first select Sun Java System AppServer as an External Application to use in your Connectivity Map (see Figure 34).

Figure 34 Connectivity Map - External Applications


Connectivity Map Editor
File External Application
Scheduler
Sun Java System AppServer External Application

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

2 Select the external systems needed for your project (for this sample, the **Sun Java System AppServer** and **File External Applications**). Icons representing the selected external systems are added to the Connectivity Map toolbar.

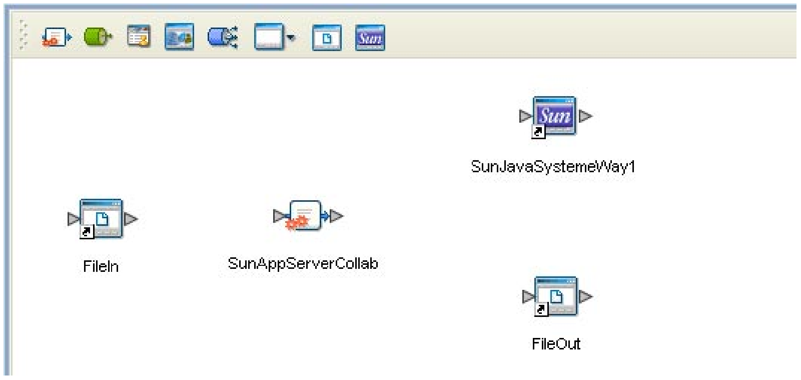
Populating the Connectivity Map

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

1 For this sample, drag the following components onto the Connectivity Map canvas as displayed in Figure 35:

- **File External System** (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 AppServer External System** (1 for this sample)

Figure 35 Connectivity Map with Components



2 Rename the objects by right-clicking the object, selecting **Rename** from the shortcut menu, and typing in the new name. Change the names as follows:

- **File1** External Application to **FileIn**
- **File2** External Application to **FileOut**
- **Service1** Service to **SunAppServerCollab** (see Figure 35).

3 Save your current changes to the Repository.

8.5.3. 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 files (for example, bmp-simpleClient.jar), provided with the Sun Java System Application Server Samples Bundle (for example, bmp-simpleClient.jar), downloadable from http://java.sun.com. Make sure that this sample bundle is installed prior to manually creating the sample.

1 From the Project Explorer tree, right-click the **SunAppServer_JCE_Sample** project and select **New** > **Object Type Definition** from the shortcut menu. The **Object Type Definition Wizard** appears.

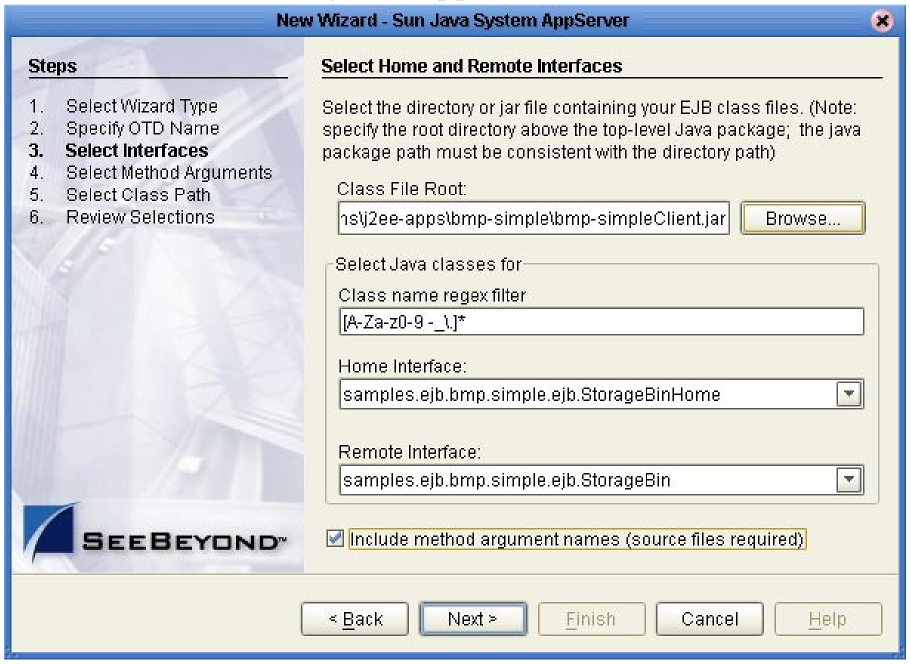
2 From the Select Wizard Type box, select **Sun Java System AppServer Wizard** and click **Next**.

3 For **step 2** of the wizard (**Specify OTD Name**), enter **Storage** as the OTD Name, enter **ejb/MyStorageBin** as the Default JNDI Name, and click Next.

4 For **step 3** of the wizard (**Select Interfaces**), browse to the location of the Sun Java System Application Server sample JAR file, **bmp-simpleClient.jar**. Click **Select** to add the file to the Selected Files box.

5 Select **Include method argument names** (see Figure 36), and click **Next**.

Figure 36 Sun Java System AppServer OTD Wizard - Step 3



Select Home and Remote Interfaces

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)

Class File Root: ns\j2ee-apps\bmp-simple\bmp-simpleClient.jar

Select Java classes for

Class name regex filter: [A-Za-z0-9 -_\.]*

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

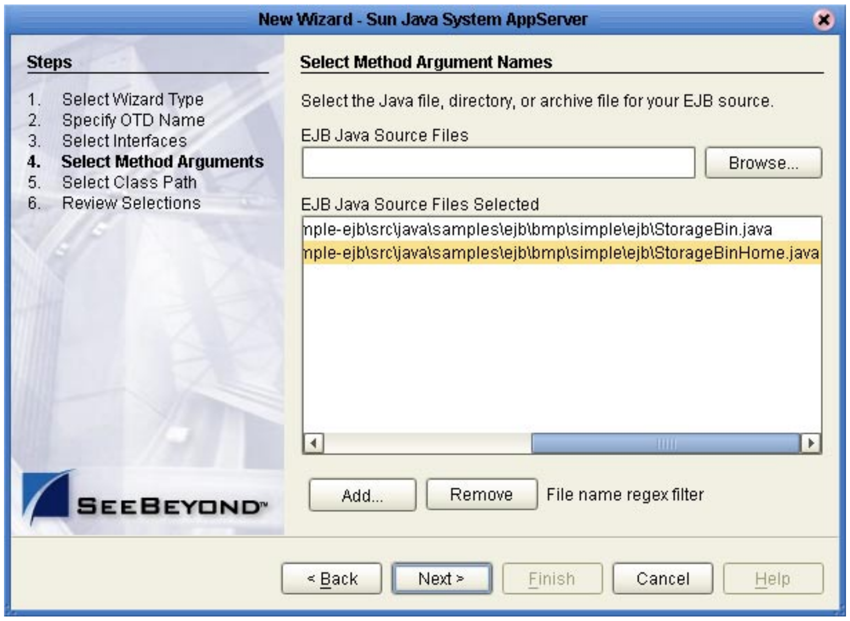
Remote Interface: samples.ejb.bmp.simple.ejb.StorageBin

☑ Include method argument names (source files required)

6 The wizard proceeds to step 5, **Select Method Arguments**. From the EJB Java Source Files field, browse to and select **StorageBin.java**. Locations vary depending on the sample version and system (example **C:\Sun\AppServer\samples\ejb\bmp\apps\simple\simple-ejb\src\java\samples\ejb\bmp\simple\ejb)**. The file is added to the **EJB Java Source Files Selected** field.

7 Click **Add**, and from the **Open** page, select **StorageBinHome.java**. Click **Open**. The file is added to the **EJB Java Source Files Selected** field (see Figure 37 on page 59).

Figure 37 Sun Java System AppServer OTD Wizard - Step 5



New Wizard - Sun Java System AppServer

Steps

1. Select Wizard Type
2. Specify OTD Name
3. Select Interfaces
4. **Select Method Arguments**
5. Select Class Path
6. Review Selections

Select Method Argument Names

Select the Java file, directory, or archive file for your EJB source.

EJB Java Source Files

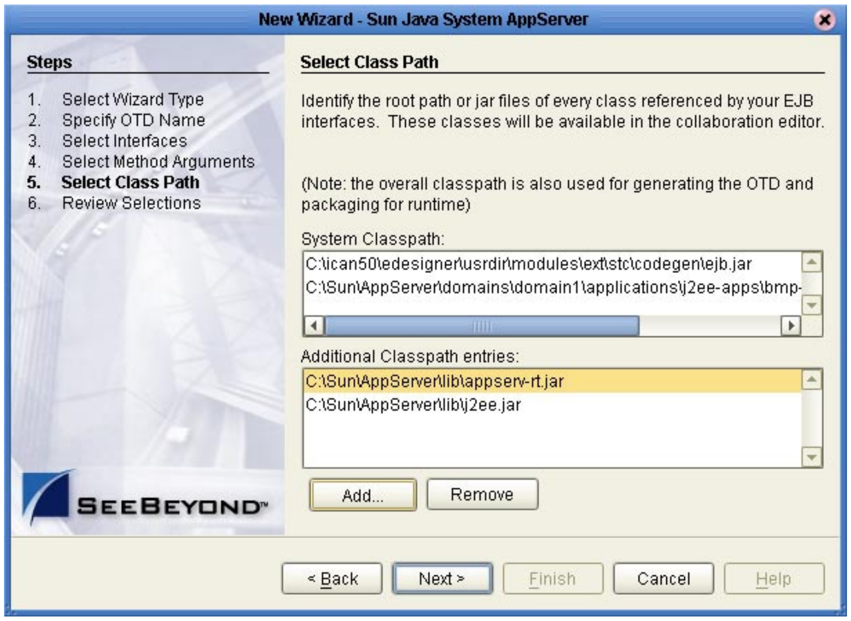
EJB Java Source Files Selected

```
nple-ejb\src\java\samples\ejb\bmp\simple\ejb\StorageBin.java
nple-ejb\src\java\samples\ejb\bmp\simple\ejb\StorageBinHome.java
```

8 Click Next, The wizard proceeds to step 6, **Select Class Path**. Click **Add**, locate and select **j2ee.jar**. The file is added to the **Additional Classpath entries** field.

9 Click the Add button again, locate and select **appserv-rt.jar** to add the file to the **Additional Classpath entries** field (see Figure 38).

Figure 38 Sun Java System AppServer OTD Wizard - Step 6



New Wizard - Sun Java System AppServer

Steps

1. Select Wizard Type
2. Specify OTD Name
3. Select Interfaces
4. Select Method Arguments
5. **Select Class Path**
6. Review Selections

Select Class Path

Identify the root path or jar files of every class referenced by your EJB interfaces. These classes will be available in the collaboration editor.

(Note: the overall classpath is also used for generating the OTD and packaging for runtime)

System Classpath:

```
C:\ican50\edesigner\usrdir\modules\ext\stc\codegen\ejb.jar
C:\Sun\AppServer\domains\domain1\applications\j2ee-apps\bmp-
```

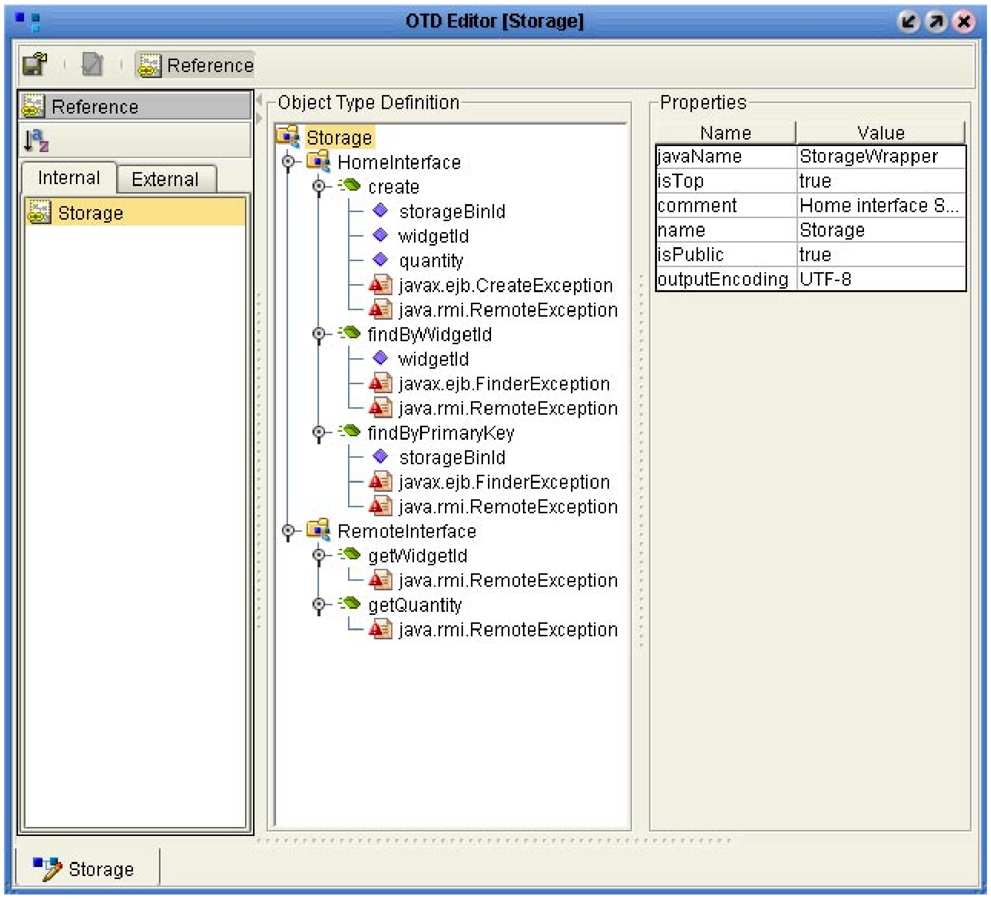
Additional Classpath entries:

```
C:\Sun\AppServer\lib\appserv-rt.jar
C:\Sun\AppServer\lib\j2ee.jar
```

10 Click **Next**. A warning appears stating that the class "com.sun.appserv.naming.S1ASCtxFactory" is not detected on the current classpath. This warning can be dismissed if load balancing will not be configured. Proceed past these warnings to the **Review Selections** step.

11 Review your selected information. When you are satisfied with your selections, click **Finish**. The OTD Editor appears with the generated **Storage** OTD (see Figure 39).

Figure 39 OTD Editor - Storage OTD



Name	Value
javaName	StorageWrapper
isTop	true
comment	Home interface S...
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 30

8.5.4. Creating Collaboration Definitions

The next step in the sample is to create the **SunAppSvr_Collab** Java Collaboration using the Collaboration Definition Wizard (Java). Once the Collaboration Definitions have been created, the Business Rules of the Collaborations are written using the Collaboration Editor (Java).

Creating the SunAppSvr_Collab Collaboration Definition

The **SunAppSvr_Collab** 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.

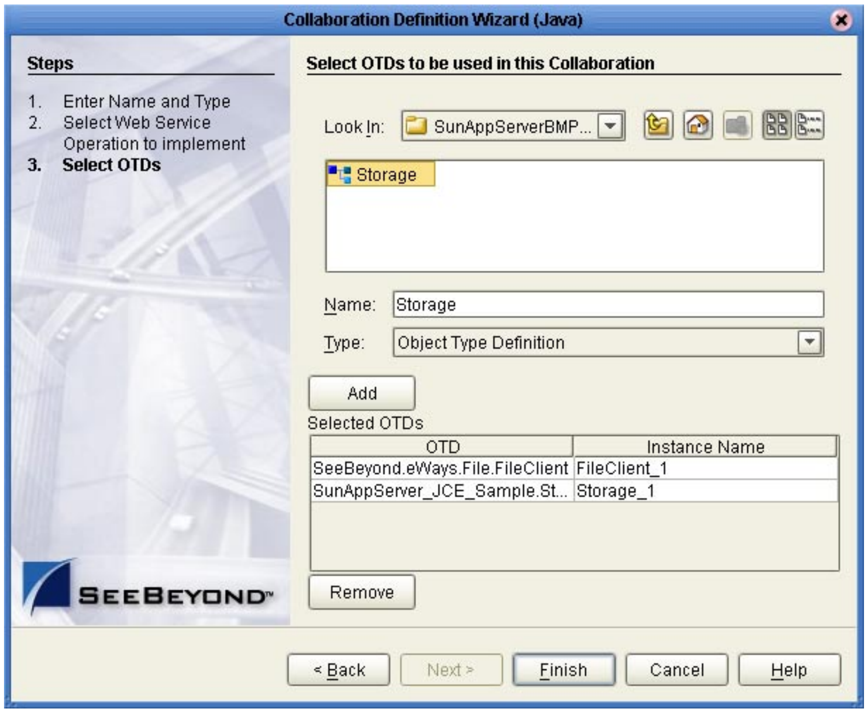
2 Enter a Collaboration Definition name (for this sample **SunAppSvr_Collab**) and click **Next**.

3 For Step 2 or the wizard, from the Web Services Interfaces selection window, double-click **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 **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 SunAppServer_JCE_Sample > Storage. The Storage OTD is added to the Selected OTDs field (see Figure 40).

Figure 40 Collaboration Definition Wizard (Java) - Select Web Service



Name: Storage
Type: Object Type Definition

OTD	Instance Name
SeeBeyond.eWays.File.FileClient	FileClient_1
SunAppServer_JCE_Sample.St...	Storage_1

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

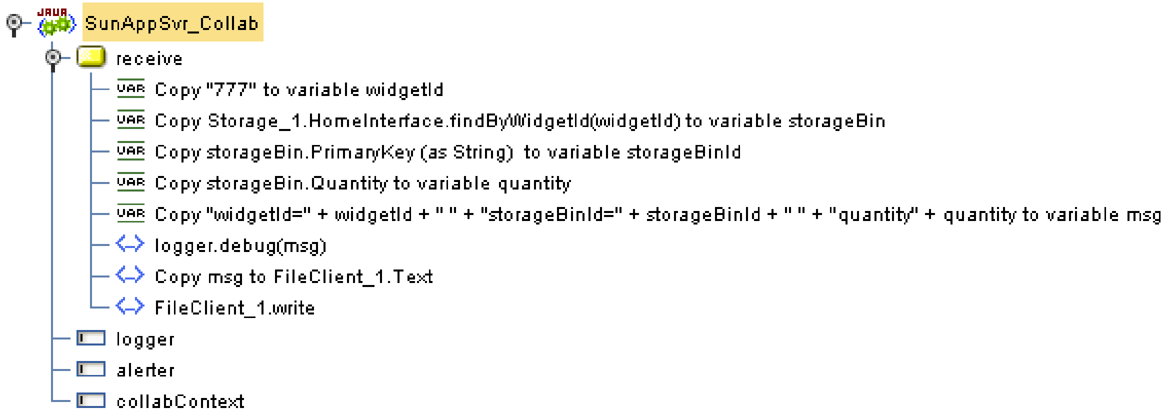
8.5.5. Using the Collaboration Editor (Java)

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

Creating the SunAppSvr_Collab Business Rules

The **SunAppSvr_Collab** Collaboration contains the Business Rules displayed in Figure 41.

Figure 41 SunAppSvr_Collab Collaboration Business Rules



```
SunAppSvr_Collab
  receive
    Copy "777" 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
```

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

1 If the Collaboration Editor is not open, double-click **SunAppSvr_Collab** in the Project Explorer tree to open the Sun **SunAppSvr_Collab** 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. Once the Comment is created, it can be moved by clicking the comment and dragging it up or down the Business Rules tree to a new location.

3 To create the **Copy "777" to variable widgetId** rule do the following:

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

B Enter **widgetId** as the variable name.

C For Type, select **Class** and click the ellipsis (...) button. The **Find Class** dialog box appears.

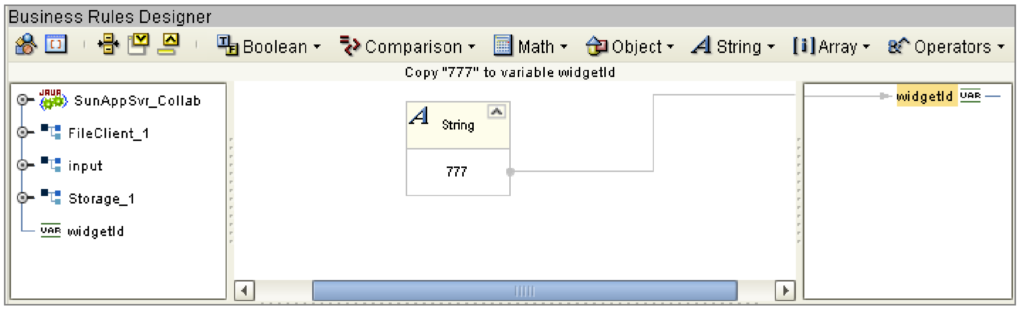
D Select **String** under **All Classes**, and click **Select**.

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

F From the Business Rules Designer **String** menu, select **Literal String**. The **String** method box appears. Enter 777 as the value.

G Map the 777 output node of the String method box to **widgetId** in the right pane of the Business Rules Designer. To do this, click on the 777 output node and drag the cursor to **widgetId** in the right pane of the Business Rules Designer. A line now connects the two nodes (see Figure 42).

Figure 42 Copy "777" to variable widgetId rule



4 To create the **Copy Storage_1.HomeInterface.findByWidgetId(widgetId) to variable storageBin** rule do the following:

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

B Enter **storageBin** as the variable name.

C For Type, select **Class** and click the ellipsis (...) button. The **Find Class** dialog box appears.

D Select **storageBin** under **All Classes**, select **getEJBHome()** under **storageBin**, and click **Select**.

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

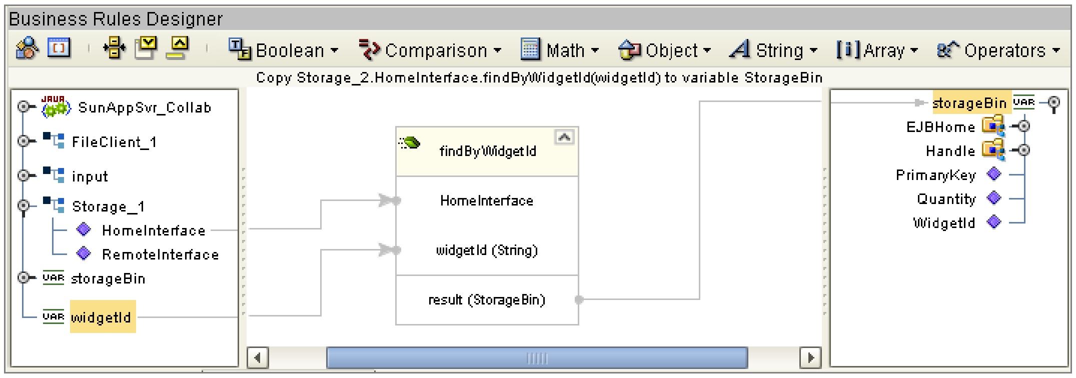
F Right-click **HomeInterface** under **Storage_1** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.

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

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

I Map the **result (storageBin)** output node of the findByWidgetId method box, to **storageBin** in the right pane of the Business Rules Designer (see Figure 43).

Figure 43 Copy Storage_1.HomeInterface.findByWidgetId(widgetId) to storageBin



5 To create the **Copy storageBin.PrimaryKey (as String) to variable storageBinId** rule do the following:

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

B Enter **storageBinId** as the variable name.

C For Type, select **Class** and click the ellipsis (...) button. The **Find Class** dialog box appears. Select **String** under **All Classes** and click **Select**.

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

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

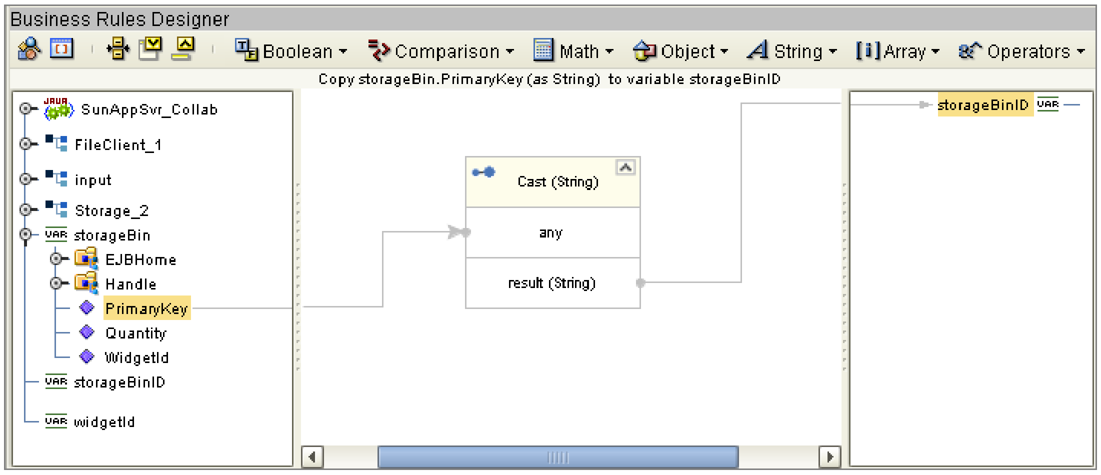
F From the Cast dialog box, select **Class** and click the ellipsis (...) button. The **Find Class** dialog box appears. Select **String** under **All Classes** and click **Select**.

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

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

I 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 44).

Figure 44 Copy storageBin.PrimaryKey (as String) to variable storageBinId rule



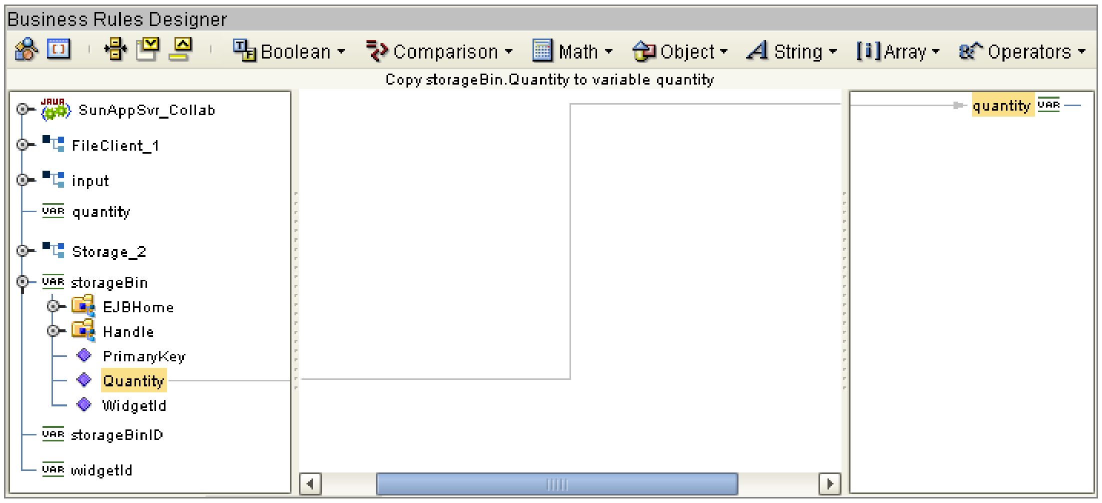
6 To create the **Copy storageBin.Quantity to variable quantity** rule do the following:

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

B Enter **quantity** as the variable name. For Type, click **Primitive:** and select **int**. Click **OK**

C Map **Quantity** under storageBin in the left pane of the Business Rules Designer, to **quantity** in the right pane of the Business Rules Designer (see**Figure 45 on page 65**).

Figure 45 Copy storageBin.Quantity to variable quantity rule



7 To create the **Copy "widgetId=" + widgetId + " " + "storageBinId=" + storageBinId + " " + "quantity" + quantity to variable msg** rule do the following:

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

B Enter **msg** as the variable name.

C For Type, select **Class** and click the ellipsis (...) button. The **Find Class** dialog box appears. Select **String** under **All Classes,** and click **Select**.

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

E From the Business Rules Designer, String menu, select **Literal String**. A **String** method box appears. Enter **widgetId=** as the value.

F Map the **widgetId=** output node of the String method box to the **value1** input node of the Add method box.

G Map **widgetId** in the left pane of the Business Rules Designer, to the **value2** input node of the Add method box. An additional input node is added to the Add method box.

H From the Business Rules Designer, String menu, select **Literal String**. A **String** method box appears. Leave the value blank.

I Map the output node of the String method box to the **value3** input node of the Add method box. An additional input node is added to the Add method box.

J From the Business Rules Designer, String menu, select **Literal String**. A **String** method box appears. Enter **storageBinId=** as the value.

K Map the **storageBinId=** output node of the String method box to the **value4** input node of the Add method box.

L Map **storageBinId** in the left pane of the Business Rules Designer, to the **value5** input node of the Add method box. An additional input node is added to the Add method box.

M From the Business Rules Designer, String menu, select **Literal String**. A **String** method box appears. Leave the value blank.

N Map the output node of the String method box to the **value6** input node of the Add method box.

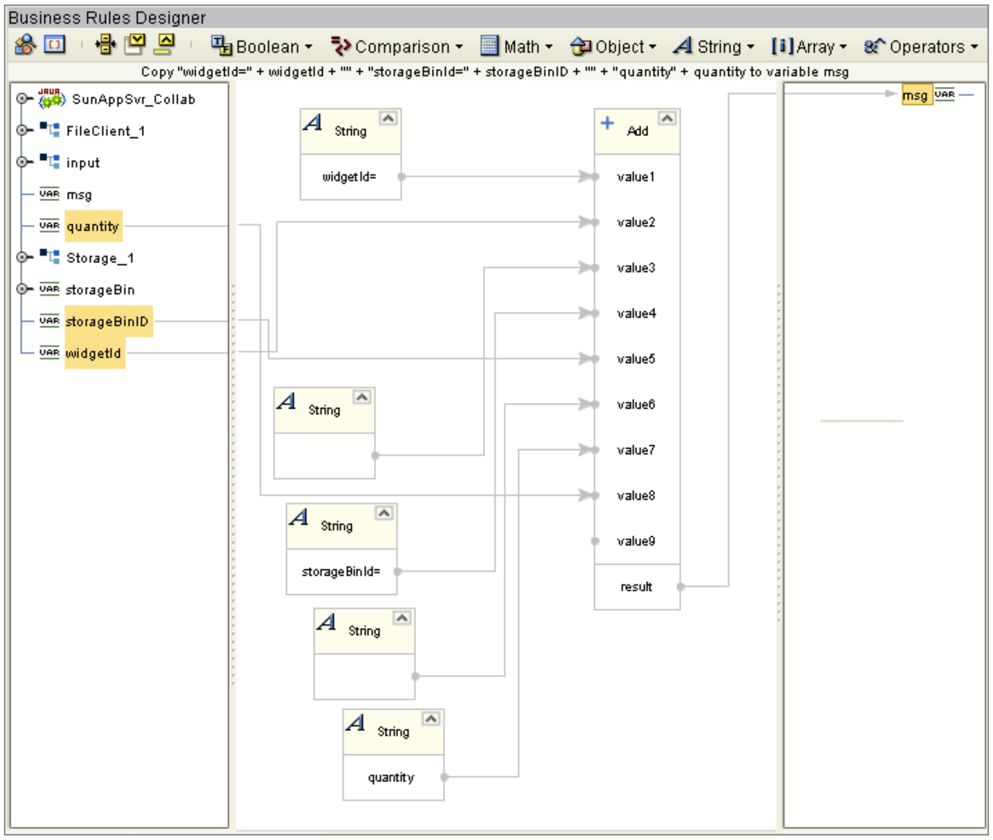
O From the Business Rules Designer, String menu, select **Literal String**. A **String** method box appears. Enter **quantity** as the value.

P Map the **quantity** output node of the String method box to the **value7** input node of the Add method box.

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

R Map the **result** output node of the Add method box to **msg** in the right pane of the Business Rules Designer (see Figure 46).

Figure 46 Collaboration Editor - SunAppSvr_Collab



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

8 To create the **logger.debug(msg)** rule do the following:

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

B Right-click **logger** (the logger field) under SunAppSvr_Collab 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 To create the **Copy msg to FileClient_1.Text** rule do the following:

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.

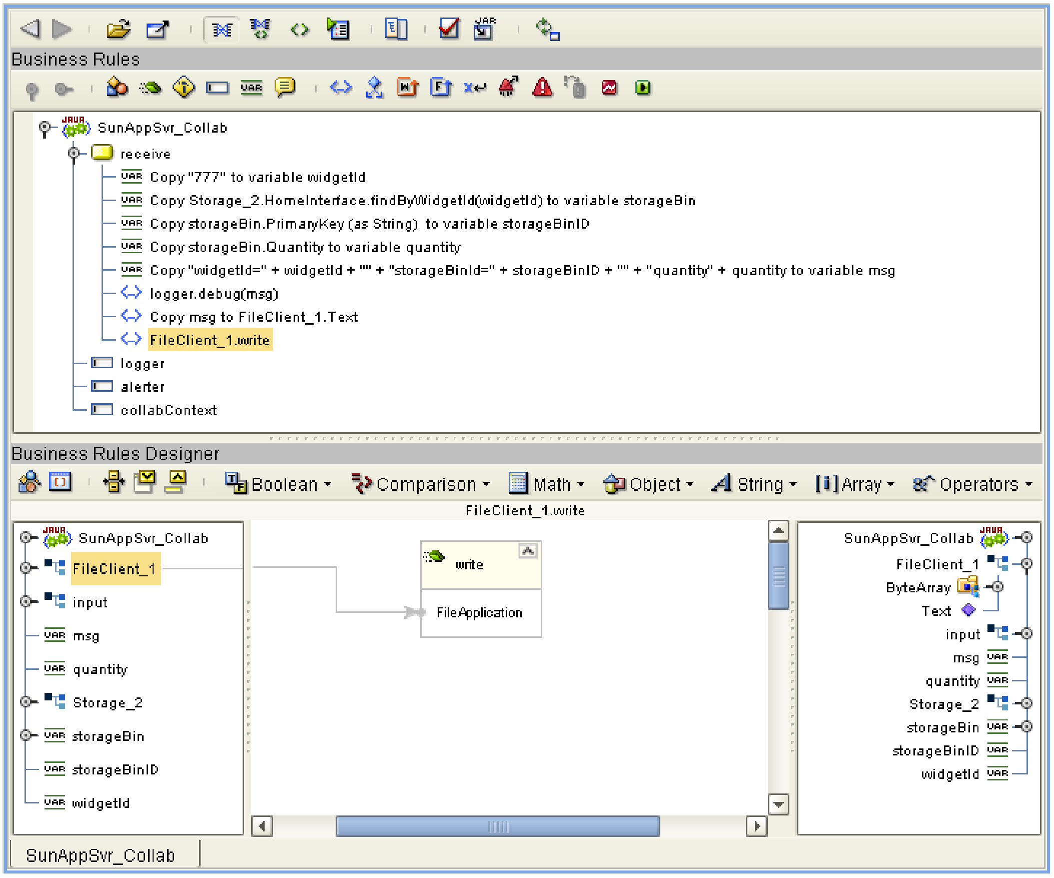
10 To create the **FileClient_1.write** rule do the following:

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 (see Figure 47).

Figure 47 Collaboration Editor - SunAppSvr_Collab



```
SunAppSvr_Collab
  receive
    Copy "777" to variable widgetId
    Copy Storage_2.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
```

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 *eGate Integrator User's Guide*.

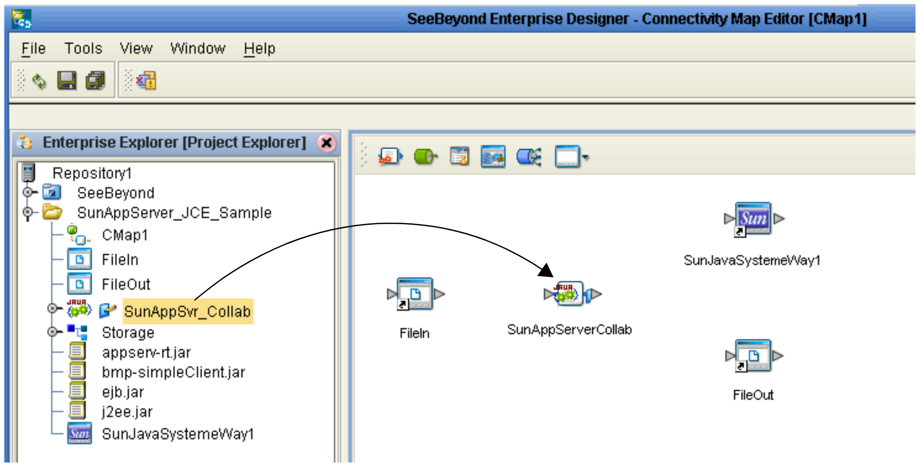
8.5.6. Binding the eWay Components

After the Collaborations have been written, the components are associated and Bindings are created in the Connectivity Map.

1 From the Project Explorer, double-click **CMap1** to display the Connectivity Map.

2 Drag and drop the **SunAppSvr_Collab** Collaboration from the Project Explorer to the **SunAppServerCollab** Service. If the Collaboration is successfully associated, the Service's "gears" icon changes from red to green (see Figure 48).

Figure 48 Connectivity Map - Associating the Project's Components



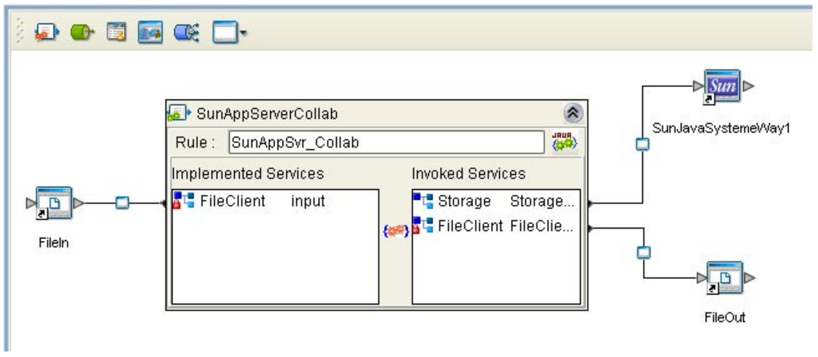
3 From the Connectivity Map canvas, double-click **SunAppServerCollab**. The **SunAppServerCollab** binding dialog box appears with the **SunAppSvr_Collab** Rule.

4 From the **SunAppServerCollab** dialog box, map **FileClient Input** (under Implemented Services) to the inbound **FileIn** External Application.

5 From the **SunAppServerCollab** dialog box, map **Storage** (under Invoked Services) to the **SunJavaSystemeWay1** External Application.

6 From the **SunAppServerCollab** dialog box, map **Storage** (under Invoked Services) to the **SunJavaSystemeWay1** External Application (see Figure 49).

Figure 49 Connectivity Map - Binding the Project's Components



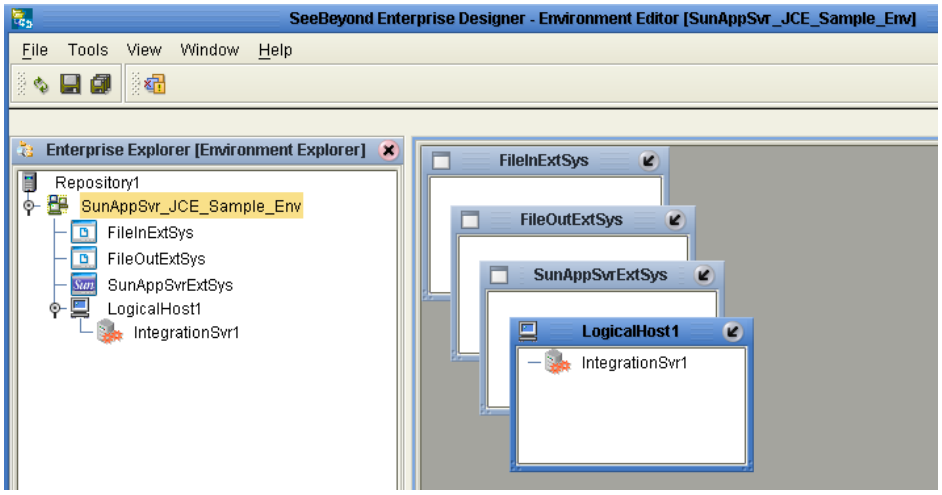
7 Minimize the **SunAppServerCollab** binding dialog box by clicking the chevrons in the upper-right corner.

8.5.7. Creating an Environment

Environments include a project's external systems, Logical Hosts, integration servers and message servers, and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Explorer and Environment Editor. For the **SunAppServer_JCE_Sample,** do the following:

1. From the Enterprise Designer's 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 **SunAppServer_JCE_Sample_Env**.
4. Right-click **SunAppServer_JCE_Sample_Env** and select **New Sun Java System AppServer External System**. Name the External System **SunAppSvrExtSys**. **SunAppSvrExtSys** is added to the Environment Editor.
5. Right-click **SunAppServer_JCE_Sample_Env** and select **New File External System**. From the **Create an External System** dialog box, enter **FileInExtSys** as the name and select **Inbound File eWay** as the type. Click **OK**. **FileInExtSys** is added to the Environment Editor.
6. Right-click **SunAppServer_JCE_Sample_Env** and select **New File External System** again. Enter **FileOutExtSys** as the name and select **Outbound File eWay** as the type. **FileOutExtSys** is added to the Environment Editor.
7. Right-click **SunAppServer_JCE_Sample_Env** and select **New Logical Host**. The **LogicalHost1** box is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.
8. From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server** from the shortcut menu. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost1 (see Figure 50).

Figure 50 Environment Editor



9. Save your current changes to the Repository.

8.5.8. Configuring the eWays

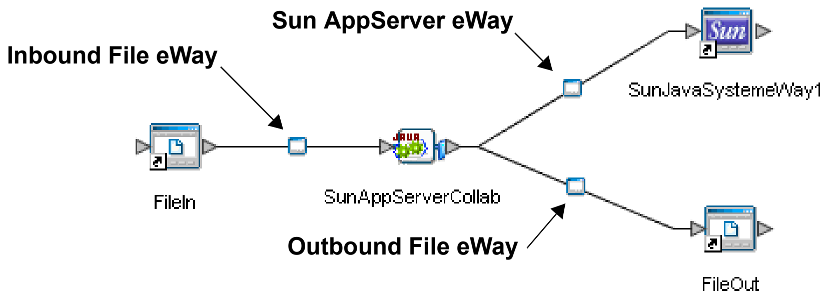
The SunAppServer_JCE_Sample project uses three eWays, each represented in the Connectivity Map as a node between an External Application and a Service. eWays facilitate communication and movement of data between the external applications and the eGate system.

The File eWay properties are configured from the Connectivity Map. The Sun AppServer eWay properties are set from both the Project Explorer's Connectivity Map and the Environment Explorer tree.

Configuring the File eWay Properties

1 Double-click the inbound **FileIn eWay** (see Figure 51), select **Inbound File eWay** in the Templates dialog box and click **OK**.

Figure 51 eWays



2 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 Connection Parameters	
Directory	C:/temp
Input file name	Input*.txt
Polling interval	5000

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

Table 6 Outbound File eWay Settings

Outbound eWay Connection Parameters	
Directory	C:/temp
Output file name	output%d.dat

Configuring the Sun AppServer eWay Properties

The Sun AppServer eWay properties are set in both the Connectivity Map and the Environment Explorer. In most cases, the default configuration is sufficient and no change is necessary. For more information on the Sun AppServer eWay properties and the Properties Editor, see **Configuring the Sun AppServer eWay Properties** on page 19 or see the *eGate Integrator User's Guide*.

Modifying 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 7, and click **OK**.

Table 7 Sun AppServer eWay Connectivity Map Properties

Inbound Sun AppServer eWay (Connectivity Map) Properties	
JNDI InitialContext Settings Set as directed, otherwise use the default settings	
JNDI name	ejb/MyStorageBin *(for example)*

Modifying the Sun AppServer eWay Environment Explorer Properties

1 From the **Environment Explorer** tree, right-click the Sun AppServer eWay External System (**SunAppSvrExtSys** 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. For this sample, the default settings should be adequate.

8.5.9 Creating and Activating 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.

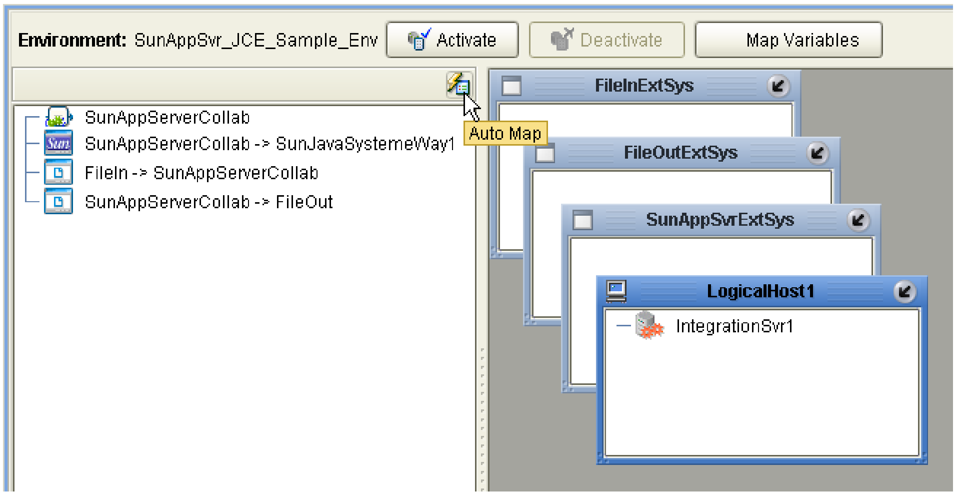
To create a Deployment Profile (using the SunAppSvr_JCE_Sample), do the following:

1 From the Enterprise Explorer's Project Explorer, right-click the project (for example, **SunAppServer_JCE_Sample**) and select **New** > **Deployment Profile**.

2 From the **Create Deployment Profile** dialog box, enter a name for the Deployment Profile (for example, **SunAppServer_JCE_Sample_DP**), select the appropriate Environment, and click **OK**.

3 Click the **Auto Map** icon as displayed in **Figure 52 on page 72**. The projects components are automatically mapped to their system window as seen in **Figure 53 on page 72**. If any of the project components are not mapped automatically after Auto Map is used, those component can be mapped manually by following the appropriate steps below. Once all components are mapped, proceed to step 8.

Figure 52 Deployment Profile - Auto Map


Environment: SunAppSvr_JCE_Sample_Env
Activate
Deactivate
Map Variables
SunAppServerCollab
SunAppServerCollab -> SunJavaSystemeWay1
FileIn -> SunAppServerCollab
SunAppServerCollab -> FileOut
Auto Map
FileInExtSys
FileOutExtSys
SunAppSvrExtSys
LogicalHost1
IntegrationSvr1

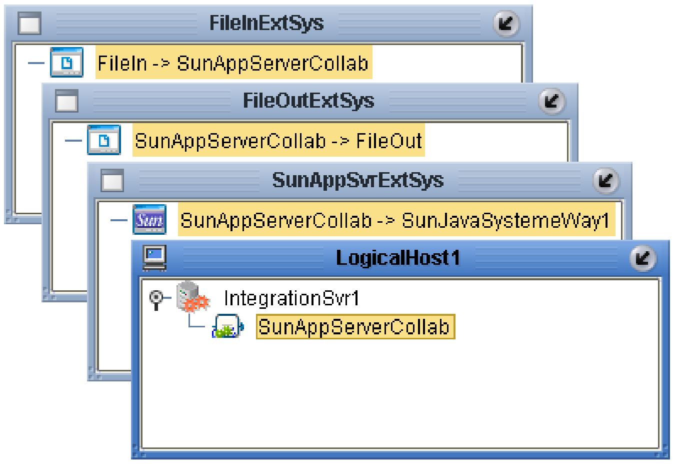
4 Drag **SunAppServerCollab** (service) to **IntegrationSvr1** in the **LogicalHost1** window.

5 From the left pane of the Deployment Editor, drag **FileIn -> SunAppServerCollab** to the **FileInExtSys** window.

6 From the left pane of the Deployment Editor, drag **SunAppServerCollab -> SunAppServerCollab** to the **SunAppSvrExtSys** window.

7 From the left pane of the Deployment Editor, drag **SunAppServerCollab -> FileOut** to the **FileOutExtSys** window (see Figure 53).

Figure 53 Deployment Profile


FileInExtSys
FileIn -> SunAppServerCollab
FileOutExtSys
SunAppServerCollab -> FileOut
SunAppSvrExtSys
SunAppServerCollab -> SunJavaSystemeWay1
LogicalHost1
IntegrationSvr1
SunAppServerCollab

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

8.5.10.Running the Project

The following directions assume that the Enterprise Designer was downloaded to **C:\ican50**. If this is not the case, replace that location in the following directions with the appropriate location.

1 From the Enterprise Manager Downloads tab, download **Logical Host - for win32**.

2 Extract the file to the **ican50\LogicalHost2** directory. You must specify the **LogicalHost2** directory for it to be created.

3 Navigate to **C:\ican50\LogicalHost2\logicalhost\bootstrap\config** directory and open the **logical-host.properties** file using a text editor.

4 Enter the following information in the appropriate fields:

- Logical Host root directory: **ican50\LogicalHost2\logicalhost**
- Repository URL: **http://localhost:*port number/repository name***
- Repository user name and password: ***Your user name and password***
- Logical Host Environment name: **SunAppServer_JCE_Sample_Env**
- Logical Host name: **LogicalHost2**

Save your changes to **logical-host.properties** and close the file.

5 Run the **bootstrap.bat** file in the **ican50\LogicalHost2\logicalhost\bootstrap\bin** directory.

6 Copy the sample input data file to the input directory.

Index