BEA WebLogic Adapter for CORBA™

User Guide

Release 7.0
Document Date: October 2002

Copyright

Copyright © 2002 BEA Systems, Inc. All Rights Reserved.

Copyright © 2002 iWay Software. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

BEA WebLogic Adapter for CORBA User Guide

Part Number	Date
N/A	October 2002

Table of Contents

About This Document

The *BEA WebLogic Adapter for CORBA User Guide* is organized as follows:

- Chapter 1, “Introducing the BEA WebLogic Adapter for CORBA,” introduces the BEA WebLogic Adapter for CORBA, describes its features, and presents an overview of how it works.
- Chapter 2, “Using the BEA Application Explorer With CORBA,” describes how to use the BEA Application Explorer to ceate service schemas.
- Chapter 3, “Creating an Application View for a CORBA Object,” describes how to create application views and add services to them.
- Appendix A, “Using CORBA Implementations With the Adapter,” provides details for Orbix2000, VisiBroker for Java, and JacORB.
- Appendix B, “Sample Files,” describe the sample files provided with the adapter.

What You Need to Know

This document is written for system integrators who develop client interfaces between CORBA and other applications. It describes how to use the BEA WebLogic Adapter for CORBA and how to develop application environments with specific focus on message integration. It is assumed that readers know Web technologies and have a general understanding of Microsoft Windows and UNIX systems.

Related Information

The following documents provide additional information for the associated software components:

- *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*
- *BEA WebLogic Adapter for CORBA Release Notes*
- *BEA Application Explorer Installation and Configuration Guide*
- BEA WebLogic Server installation and user documentation, which is available at the following URL:

  `http://edocs.bea.com/more_wls.html`
- BEA WebLogic Integration installation and user documentation, which is available at the following URL:

  `http://edocs.bea.com/more_wli.html`

Contact Us!

Your feedback on the BEA WebLogic Adapter for CORBA documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the BEA WebLogic Adapter for CORBA documentation.

In your e-mail message, please indicate which version of the BEA WebLogic Adapter for CORBA documentation you are using.

If you have any questions about this version of the BEA WebLogic Adapter for CORBA, or if you have problems using the BEA WebLogic Adapter for CORBA, contact BEA Customer Support through BEA WebSupport at www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`

Convention	Item
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Examples*: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Introducing the BEA WebLogic Adapter for CORBA

This section introduces the BEA WebLogic Adapter for CORBA, describes its features, and provides an overview of how it works. It includes the following topics:

- Introduction
- How the BEA WebLogic Adapter for CORBA Works

Introduction

A number of companies and application providers have used Common Object Request Broker Architecture (CORBA) for internet and legacy C++ application development, particularly before the popularity of Java or J2EE and XML or Web Service architectures. The BEA WebLogic Adapter for CORBA integrates existing CORBA services with WebLogic Server and WebLogic Integration so that existing IT investments can be integrated into J2EE applications and deployed as Web services.

The BEA WebLogic Adapter for CORBA allows CORBA-based applications on WebLogic Server and WebLogic Integration to communicate with other applications integrated by the WebLogic Integration adapter suite. Access to CORBA environments is provided through the adapter, which uses CORBA Interface

Definition Language (IDL) entries to generate local, remote-based services. Applications make calls to a remote service that, in turn, invokes a CORBA method and receives return information from the CORBA method.

A CORBA object provides distributed object capability between applications in a network. Although a CORBA object is implemented using a standard programming language, each CORBA object has a clearly defined interface, defined using the CORBA Interface Definition Language (IDL). The definition of a CORBA object is consistent with the definition presented by the Object Management Group (OMG). OMG has a number of specifications and documents that provide complete details on objects. For more information, visit the OMG web site, located at http://www.omg.org.

The BEA WebLogic Adapter for CORBA provides a means to exchange real-time business data between CORBA servers and other application, database, or external business partner systems. The adapter allows for inbound and outbound processing with CORBA.

The adapter uses WebLogic Integration and XML messages to allow non-CORBA applications to communicate and exchange transactions with CORBA. Applications that need to access CORBA data when a CORBA business event occurs use the adapter in combination with WebLogic Integration application views, events, and business process workflows to receive messages from CORBA. Applications that need to cause a CORBA business event use the adapter in combination with WebLogic Integration application views, services, and business process workflows to send request messages to CORBA. If the request is for retrieving data from CORBA, then the adapter sends the application a response message with the data.

The BEA WebLogic Adapter for CORBA provides these key features:

- *Support for several ORBs*, such as JacORB, Orbix2000, and VisiBroker for Java.
- *Guaranteed synchronous and asynchronous bi-directional message interactions* between WebLogic Integration and an object request broker (ORB).
- *Data transfer* between a business process running within WebLogic Integration and an ORB.
- *Service adapter integration operations* providing end-to-end business process management using XML schemas.
- *BEA Application Explorer*, which uses CORBA Object Manager metadata to build XML schemas for application view services.

How the BEA WebLogic Adapter for CORBA Works

When the BEA WebLogic Adapter for CORBA receives a request for a service, the adapter converts the XML request document to a Dynamic Invocation Interface (DII) call and sends it to the CORBA server using the Internet Inter-ORB Protocol (IIOP). The response of the CORBA server is returned over IIOP. The adapter then formats the response as an XML response document for WebLogic Integration.

To create an application view from a CORBA server, you provide the adapter with configuration information for an Interoperable Naming Service (INS) and Interface Repository (IFR). The BEA Application Explorer displays a list of available systems. You can expand a system to display available servers and interfaces that were specified using CORBA IDL. You can expand an interface and create XSD service schemas, which are exported to WebLogic Integration as application views.

To communicate with the Naming Service and the Interface Repository, the adapter uses CORBA APIs to retrieve Interface Object Repositories the first time they are needed. When the application view schema is created, BEA Application Explorer uses the adapter to retrieve the server's definition from the Interface Repository. The information is converted into XML schemas for use by WebLogic Integration. If a server's interface changes, you must refresh the definitions that the adapter is using in order to avoid an out-of-date adapter run-time call, which would cause the call to fail if the new interface is incompatible with the old one. To refresh the definitions, stop the adapter processes from the WebLogic Integration console, and reload the new definitions the next time you view the system in the BEA Application Explorer.

CHAPTER

2 Using the BEA Application Explorer With CORBA

This section describes how to connect to an Object Request Broker and create service schemas. It includes the following topics:

- Overview
- Connecting to an Object Request Broker
- Creating Service Schemas

Overview

The BEA Application Explorer supports the creation of schemas based on specific tables and resulting answer sets. To obtain metadata about the Object Request Broker (ORB), the BEA Application Explorer connects to the Interface Repository. The BEA WebLogic Adapter for CORBA extracts the definition of CORBA servers and converts them to XML schemas and service XML request and response definitions. You can see the original definitions of the CORBA servers using the BEA Application Explorer.

The BEA Application Explorer displays a tree of all defined systems. By drilling down into the tree, you can determine which CORBA servers are available and how their interfaces will appear. Before you can view a CORBA server, you must ensure that the server is registered in the Naming Service and its IDL is loaded into the Interface Repository (IFR). After creating a connection in the BEA Application Explorer, you can use the Explorer to verify that the system definition was entered correctly.

Using the BEA WebLogic Adapter for CORBA, the BEA Application Explorer populates each system folder with object data retrieved from both the Naming Service and the Interface Repository. The contents of the Naming Service and Interface Repository appear in the Interfaces folder and the Objects folder, respectively.

The Objects folder contains all the servers registered in the Naming Service. If you expand a server you can see the interface that is implemented by that server. Expanding this interface displays its methods, return arguments, and parameters. Additional folders may appear under the Objects folder if additional modules were created. The Interfaces folder displays a list of modules, appearing as folders. These IFR modules represent the different work areas of CORBA.

The XML schema defines the format of XML requests and corresponding replies to the service adapter. The schema is a language-neutral interface description in XML format that declares the types, objects, and methods for the CORBA system. Conceptually, the XML schema is the same as the CORBA IDL.

Note: Before creating schemas and application view services in the BEA WebLogic Adapter for CORBA, you can save time by verifying that your ORB infrastructure is properly configured, your server is registered in the Naming Service, and your interface repository (IFR) is running and populated.

Connecting to an Object Request Broker

To connect to an Object Request Broker (ORB):

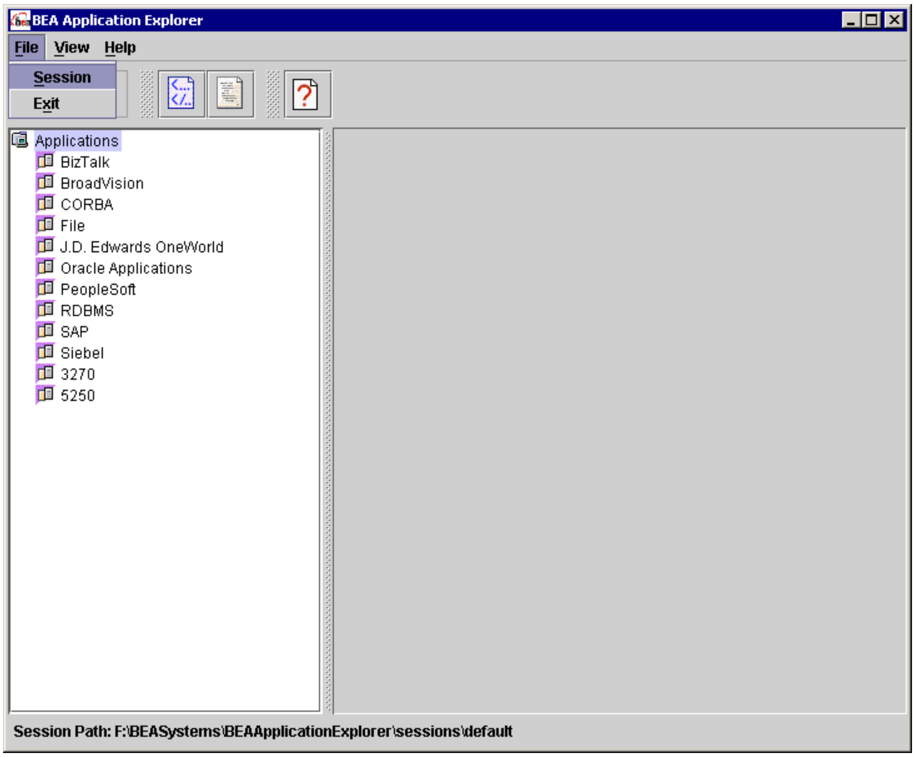
1. Start the BEA Application Explorer:

   - *For JacORB*, from the command line run `ae.bat` (on a Windows system) or `ae` (on a UNIX system) under `*installation_directory*/bin`.
   - *For all other ORBs,* choose Start→Programs→BEA Application Explorer (on a Windows system), or run shell script `ae` under `*installation_directory*/bin` (on a UNIX system).

   When you start the BEA Application Explorer, the left pane displays all the adapters supported by your version.

2. From the File menu, choose Session to change the default session path.

Figure 2-1 Choosing Session in BEA Application Explorer



Session Path: F:\BEASystems\BEAApplicationExplorer\sessions\default

The Enter Session Path window opens, displaying the default session path.

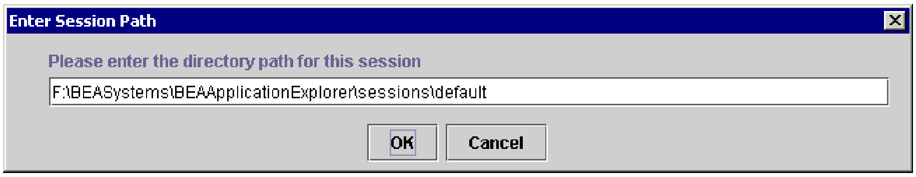
The session path holds the schemas you that generate and your ORB connection information:

`session_path\corba\connection_name\schemas`

3. If you want to accept the default session path, click OK. Otherwise, to specify a different path, enter the path.

   For example, you may want to specify a path for a particular project or for a logical grouping of services and events.

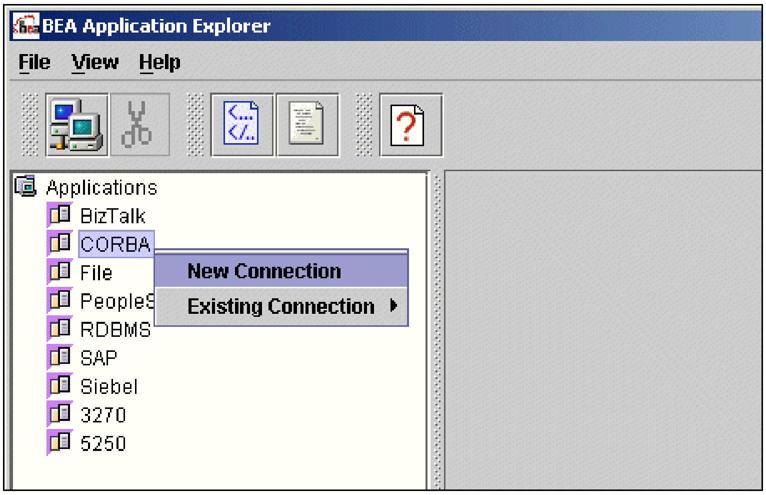
Figure 2-2 Enter Session Path Dialog Box



F:\BEASystems\BEAApplicationExplorer\sessions\default

4. You can define a new connection to an ORB or use an existing connection:
   - To define a new connection to an ORB, right-click CORBA→New Connection. A dialog box opens prompting you for a connection name; continue with step 5.
   - To use an existing connection, right click CORBA→Existing Connection→*your connection*. The connection is displayed below the CORBA node in the left pane; skip ahead to step 7.

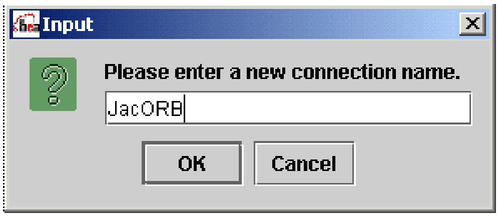
Figure 2-3 Selecting a New Connection in BEA Application Explorer



5. Enter a connection name and click OK.

   In the following figure, JacORB is the name of a logical connection.

Figure 2-4 New Connection Name Input Window



Please enter a new connection name.

JacORB

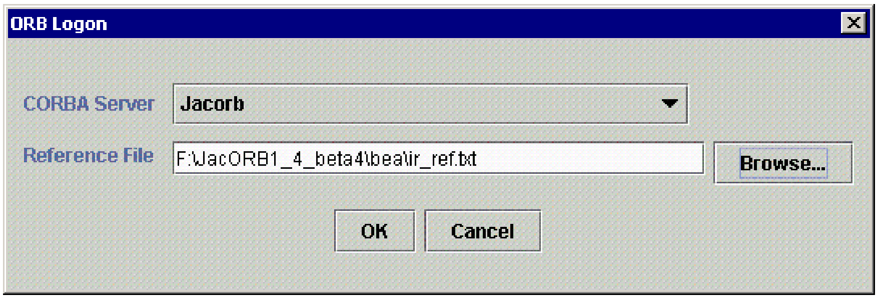
The ORB Logon window opens.

6. Select a CORBA Server and specify the name and path the Interface Repository reference file.

   This file contains the parameters required to connect to the CORBA system.

   Note: The example presented here illustrates accessing JacORB services. For other ORBs, there are additional parameters, such as Naming Service, Host, Port, and Enable IIOP Tracing. For details, see your ORB documentation.

Figure 2-5 ORB Logon Window



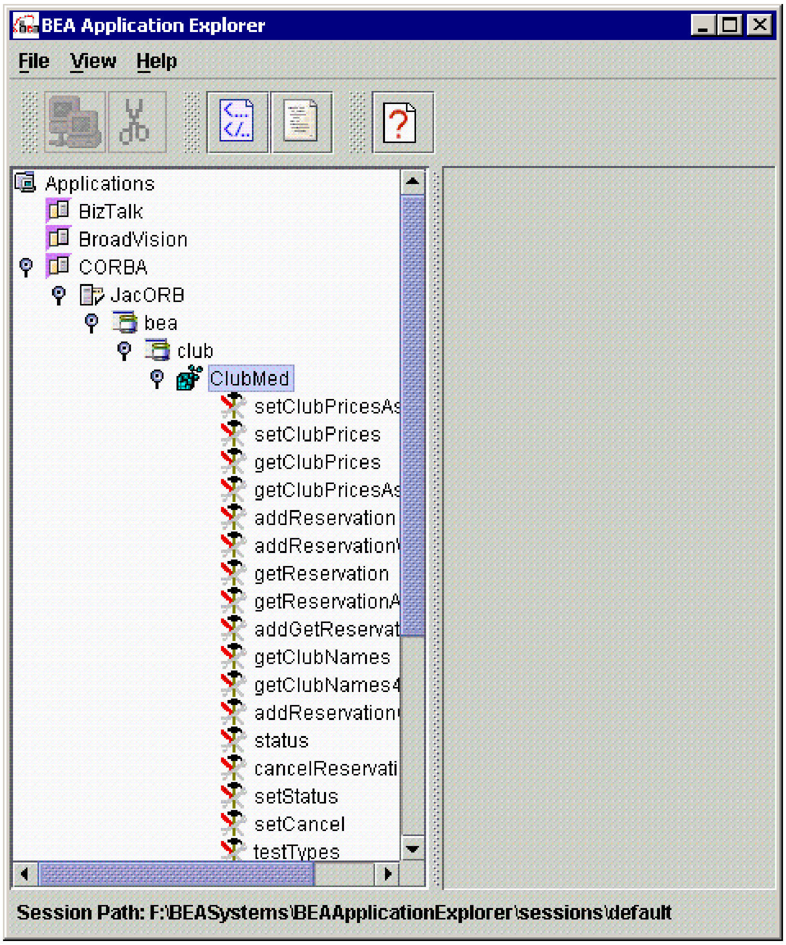
CORBA Server: Jacorb

Reference File: F:\JacORB1_4_beta4\bea\ir_ref.txt

The BEA Application Explorer loads the application information and connects to the ORB to extract the object definitions from the Interface Repository.

7. You can now expand the tree to browse the available objects. The following figure shows all available business services in a JacORB sample system called ClubMed.

Figure 2-6 Available Objects Window



Applications
BizTalk
BroadVision
CORBA
JacORB
bea
club
ClubMed
setClubPricesAs
setClubPrices
getClubPrices
getClubPricesAs
addReservation
addReservationW
getReservation
getReservationA
addGetReservat
getClubNames
getClubNames4
addReservation
status
cancelReservati
setStatus
setCancel
testTypes

Session Path: F:\BEASystems\BEAApplicationExplorer\sessions\default

For information about creating service schemas, see "Creating Service Schemas" on page 8.

Creating Service Schemas

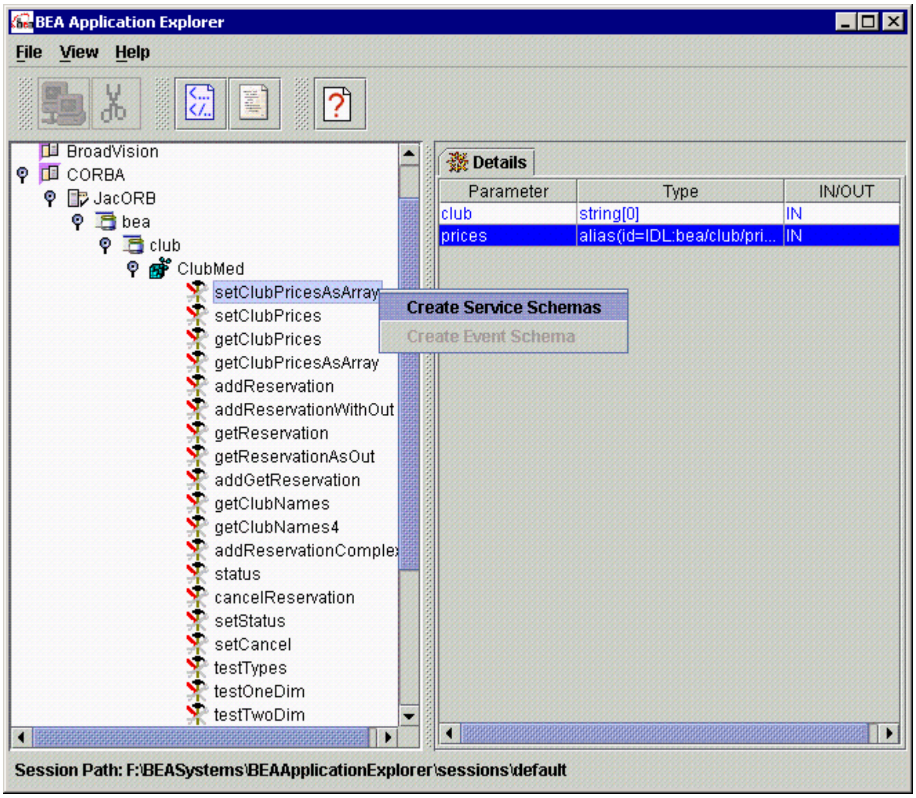
The BEA Application Explorer generates the following WebLogic Integration schemas:

- Service XML request schemas.
- Service XML response schemas.

To create service schemas:

1. Open the BEA Application Explorer and browse an object.
2. Right-click the desired service and choose Create Service Schemas.

Figure 2-7 Service Schema Creation Window



Parameter	Type	IN/OUT
club	string[0]	IN
prices	alias(id=IDL:bea/club/pri...	IN

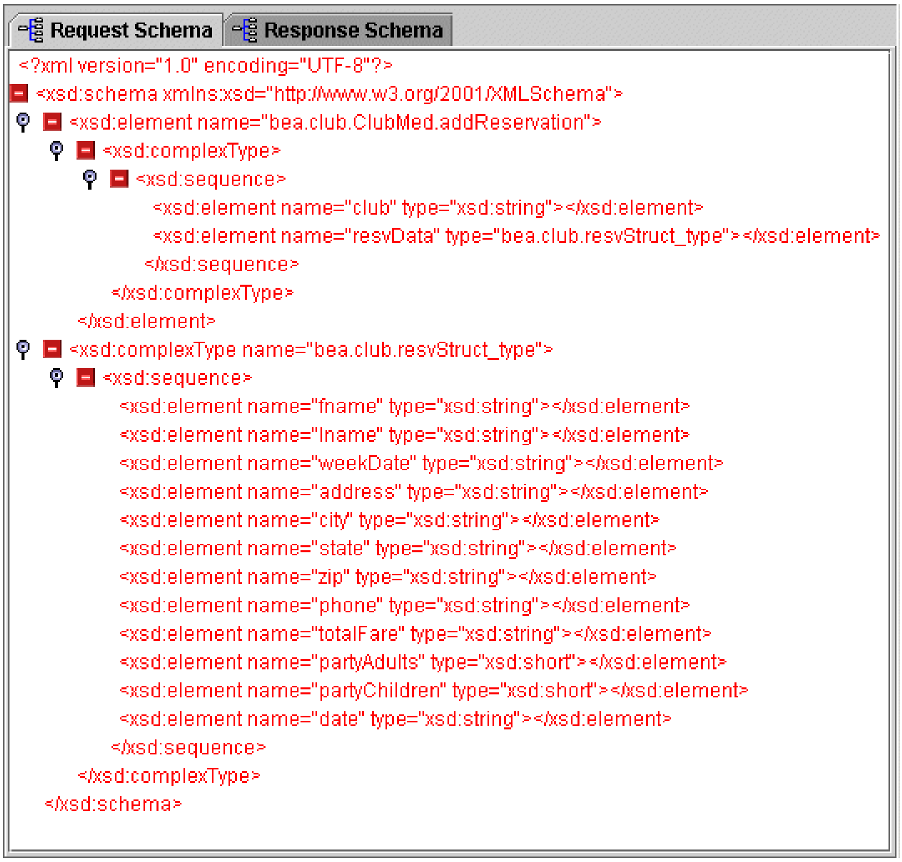
Session Path: F:\BEASystems\BEAApplicationExplorer\sessions\default

The BEA Application Explorer accesses the CORBA Interface Repository via IIOP and builds XSD schemas, which are then published to the WebLogic Integration Repository. You can view the request and response schemas for the CORBA interface object.

Request Schema and Response Schema tabs open in the right pane.

3. To view the request schema, click the Request Schema tab in the right pane.

Figure 2-8 Request Schema Tab

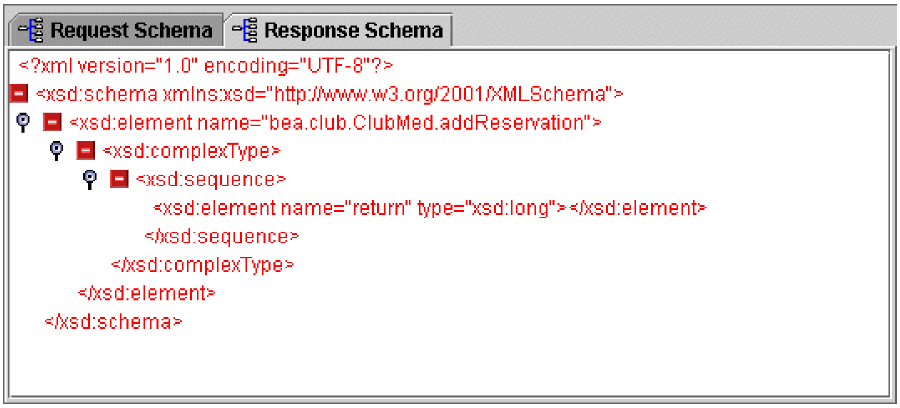


Request Schema | Response Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="bea.club.ClubMed.addReservation">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="club" type="xsd:string"></xsd:element>
        <xsd:element name="resvData" type="bea.club.resvStruct_type"></xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="bea.club.resvStruct_type">
    <xsd:sequence>
      <xsd:element name="fname" type="xsd:string"></xsd:element>
      <xsd:element name="lname" type="xsd:string"></xsd:element>
      <xsd:element name="weekDate" type="xsd:string"></xsd:element>
      <xsd:element name="address" type="xsd:string"></xsd:element>
      <xsd:element name="city" type="xsd:string"></xsd:element>
      <xsd:element name="state" type="xsd:string"></xsd:element>
      <xsd:element name="zip" type="xsd:string"></xsd:element>
      <xsd:element name="phone" type="xsd:string"></xsd:element>
      <xsd:element name="totalFare" type="xsd:string"></xsd:element>
      <xsd:element name="partyAdults" type="xsd:short"></xsd:element>
      <xsd:element name="partyChildren" type="xsd:short"></xsd:element>
      <xsd:element name="date" type="xsd:string"></xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

4. To view the response schema, click the Response Schema tab in the right pane.

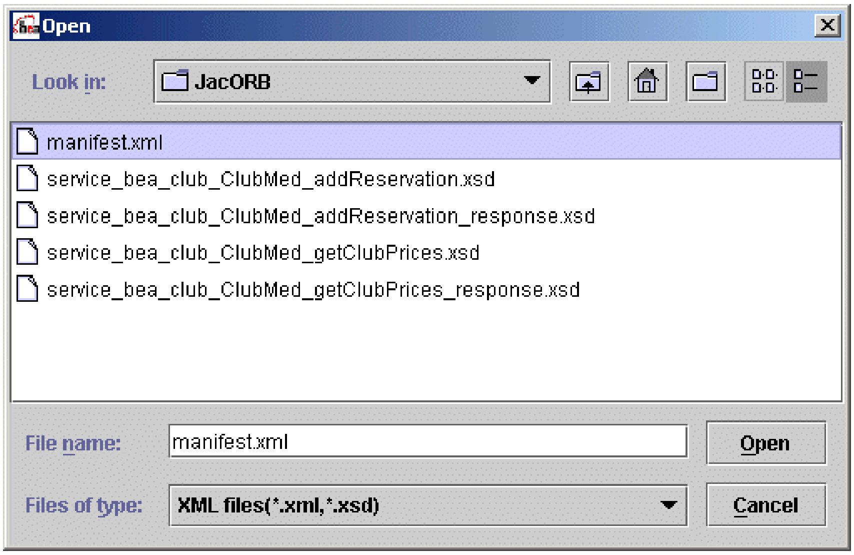
Figure 2-9 Response Schema Tab



```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="bea.club.ClubMed.addReservation">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="return" type="xsd:long"></xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

5. To view the `manifest.xml` file, choose View→View XML.

Figure 2-10 View XML Window



manifest.xml
service_bea_club_ClubMed_addReservation.xsd
service_bea_club_ClubMed_addReservation_response.xsd
service_bea_club_ClubMed_getClubPrices.xsd
service_bea_club_ClubMed_getClubPrices_response.xsd

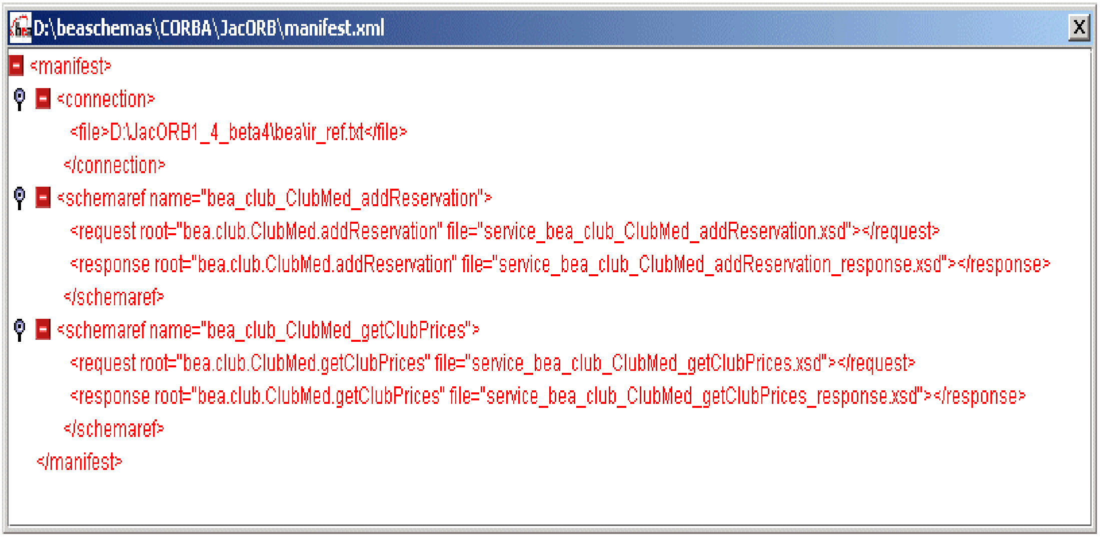
File name: manifest.xml

Files of type: XML files(*.xml,*.xsd)

6. Click `manifest.xml`.

   The `manifest.xml` file opens.

Figure 2-11 Sample manifest.xml Window



D:\beaschemas\CORBA\JacORB\manifest.xml

```
<manifest>
  <connection>
    <file>D:\JacORB1_4_beta4\bea\ir_ref.txt</file>
   </connection>
  <schemaref name="bea_club_ClubMed_addReservation">
    <request root="bea.club.ClubMed.addReservation" file="service_bea_club_ClubMed_addReservation.xsd"></request>
    <response root="bea.club.ClubMed.addReservation" file="service_bea_club_ClubMed_addReservation_response.xsd"></response>
   </schemaref>
  <schemaref name="bea_club_ClubMed_getClubPrices">
    <request root="bea.club.ClubMed.getClubPrices" file="service_bea_club_ClubMed_getClubPrices.xsd"></request>
    <response root="bea.club.ClubMed.getClubPrices" file="service_bea_club_ClubMed_getClubPrices_response.xsd"></response>
   </schemaref>
 </manifest>
```

The BEA WebLogic Adapter for CORBA uses `manifest.xml` and the service schemas to manage the interaction between an application view and the ORB.

CHAPTER

3 Creating an Application View for a CORBA Object

This section describes how to create application views and add services to them. It includes the following topics:

Overview

The BEA WebLogic Adapter for CORBA is a service adapter that is capable of processing IIOP CORBA object methods embedded in XML requests, and forwarding them to an Object Request Broker (ORB). The ORB object returns the data to the adapter, which in turn returns it to the client.

A service adapter performs the following functions:

1. Receives a service request from an external client.
2. Transforms the XML request document, which conforms to the services's request schema, into an ORB-specific format.
3. Invokes the underlying ORB function and waits for a response.
4. Transforms the response from the ORB-specific data format to an XML document that conforms to the service's response schema.

A service can be invoked asynchronously or synchronously. When a service is invoked asynchronously, the client application issues a service request and then proceeds with other processing. The client application does not wait for a response. When a service is invoked synchronously, the client waits for the response before proceeding with further processing. WebLogic Integration supports both these invocation methods, so you do not need to provide this functionality in your own application code.

Creating an Application View Folder

Application views reside within WebLogic Integration. WebLogic Integration provides you with a root folder in which you can store all of your application views; if you wish, you can create additional folders to organize related application views into groups.

To create an application view folder:

1. Log on to the WebLogic Integration Application View Console at `//appserver-host:port/wlai`.

   Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

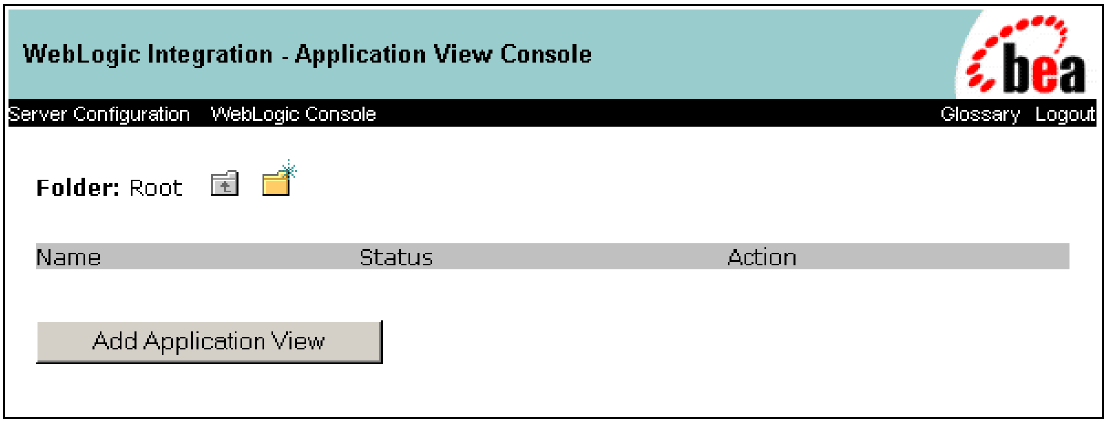
2. If prompted, enter a user name and password.

   Note: If the user name is not `system`, it must be included in the `adapter` group. For more information on adding the administrative server user name to the `adapter` group, see the *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*.

3. Click Login.

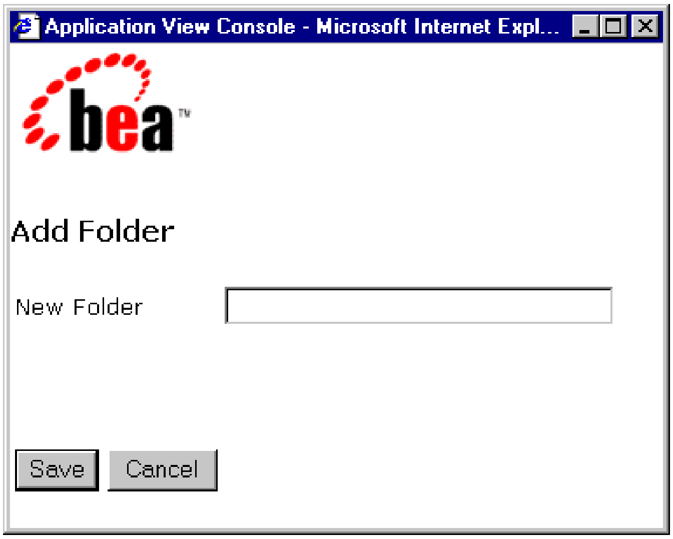
   The WebLogic Integration Application View Console opens.

Figure 3-1 Application View Console Main Window



4. Double-click the new folder icon. The Add Folder window opens.

Figure 3-2 Add Folder Window



Add Folder

New Folder

5. Supply a name for the folder, and then click Save.

You have finished creating the application view folder. To create an application view, see "Creating an Application View for a CORBA Object" on page 4.

Creating an Application View for a CORBA Object

When you define an application view, you are creating an XML-based interface between WebLogic Server and a particular CORBA Object Request Broker (ORB) within your enterprise. Once you create the application view, a business analyst can create business processes that use the application view. You can create any number of application views, each with any number of services. For more information, see "Defining an Application View" in *Using Application Integration*:

- For WebLogic Integration 7.0, see

  `http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm`

- For WebLogic Integration 2.1, see

  `http://edocs.bea.com/wlintegration/v2_1sp/aiuser/2usrdef.htm`

To create an application view:

1. Log on to the WebLogic Application View Console at `//`*`appserver-host`*`:`*`port`*`/wlai`.

   Here, *`appserver-host`* is the IP address or host name where the WebLogic Integration Server is installed, and *`port`* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

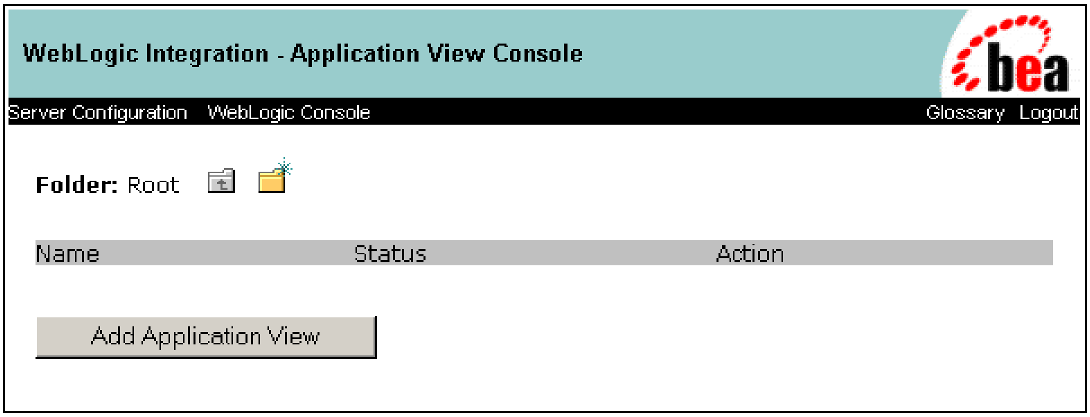
2. If prompted, enter a user name and password.

   Note: If the user name is not `system`, it must be included in the `adapter` group. For more information on adding the administrative server user name to the `adapter` group, see the *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*.

3. Click Login.

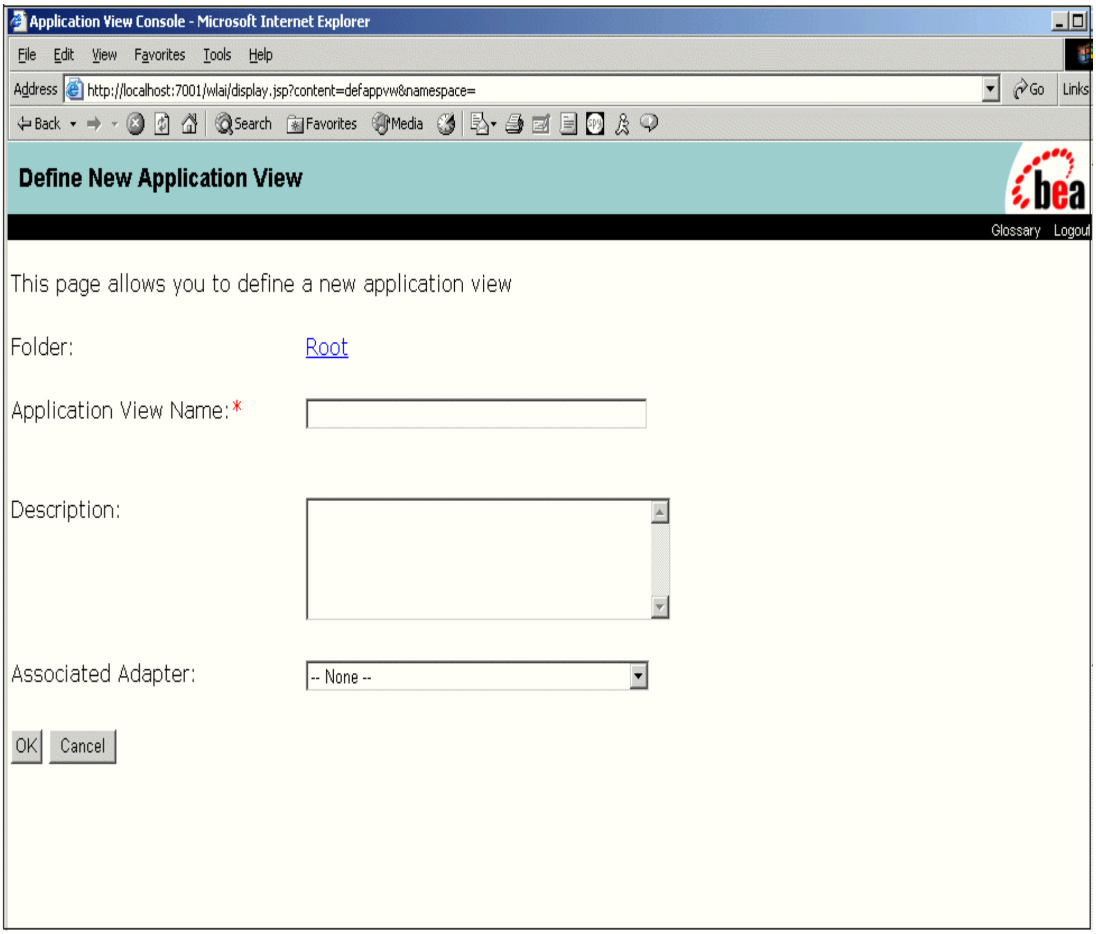
   The WebLogic Integration Application View Console opens.

Figure 3-3 Application View Console Window



4. Click Add Application View. The Define New Application View window opens.

Figure 3-4 Define New Application View Window



Application View Console - Microsoft Internet Explorer

Address: http://localhost:7001/wlai/display.jsp?content=defappvw&namespace=

Define New Application View

This page allows you to define a new application view

Folder: Root

Application View Name:*

Description:

Associated Adapter: -- None --

5. In the Application View Name field, enter a name.

   The name should describe the set of functions performed by this application. Each application view name must be unique to its adapter. Valid characters include a-z, A-Z, 0-9, and the _ (underscore) character.

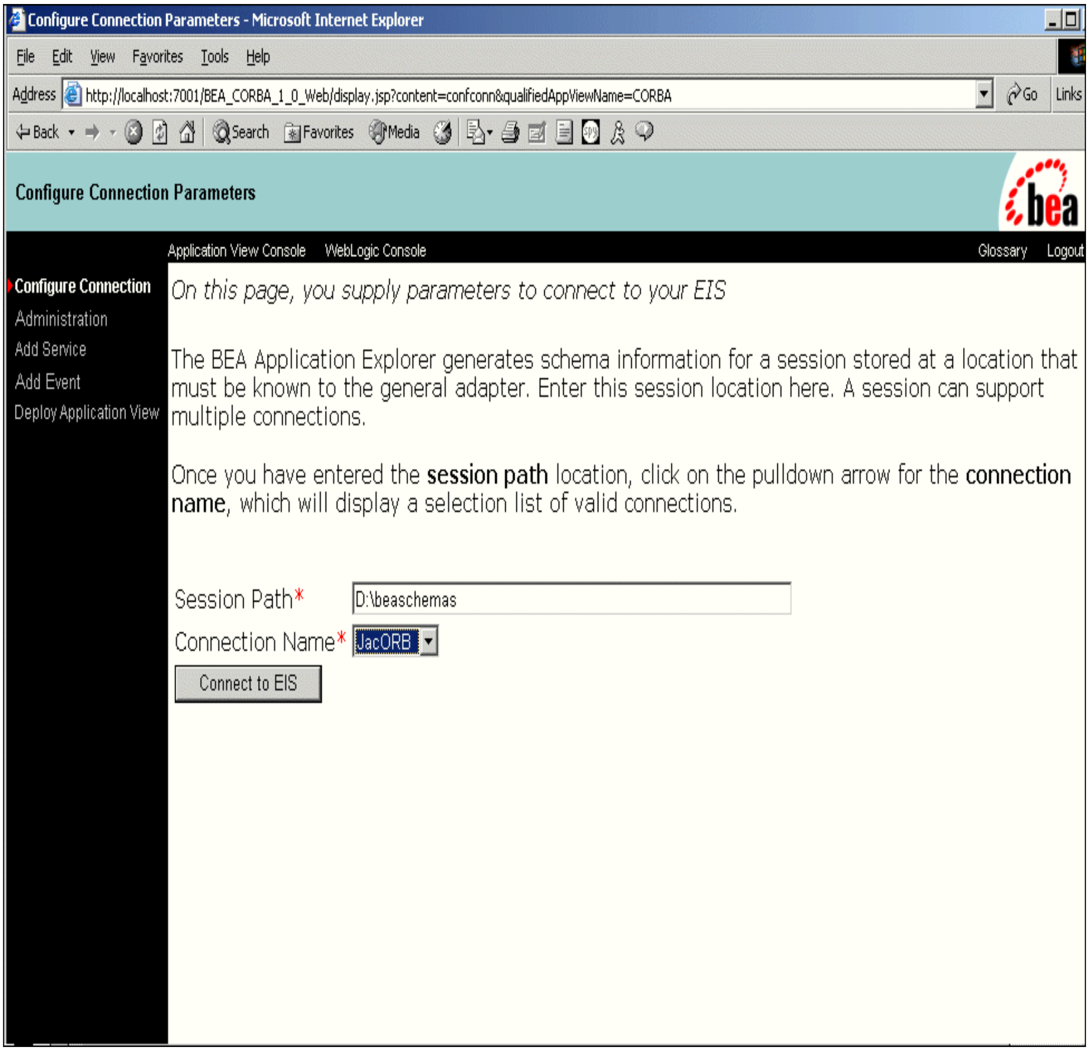
6. In the Description field, enter any relevant notes.

   These notes are viewed by users when they use this application view in workflows.

7. From the Associated Adapter drop-down list, select `BEA_CORBA_1_0.ear` (the adapter ear file).

8. Click OK. The Configure Connection Parameters window opens.

Figure 3-5 Configure Connection Parameters Window


Configure Connection Parameters - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:7001/BEA_CORBA_1_0_Web/display.jsp?content=confconn&qualifiedAppViewName=CORBA
Go Links
Back Search Favorites Media
bea
Application View Console WebLogic Console Glossary Logout
Configure Connection
Administration
Add Service
Add Event
Deploy Application View
Connect to EIS

Configure Connection Parameters

On this page, you supply parameters to connect to your EIS

The BEA Application Explorer generates schema information for a session stored at a location that must be known to the general adapter. Enter this session location here. A session can support multiple connections.

Once you have entered the **session path** location, click on the pulldown arrow for the **connection name**, which will display a selection list of valid connections.

Session Path* D:\beaschemas

Connection Name* JacORB

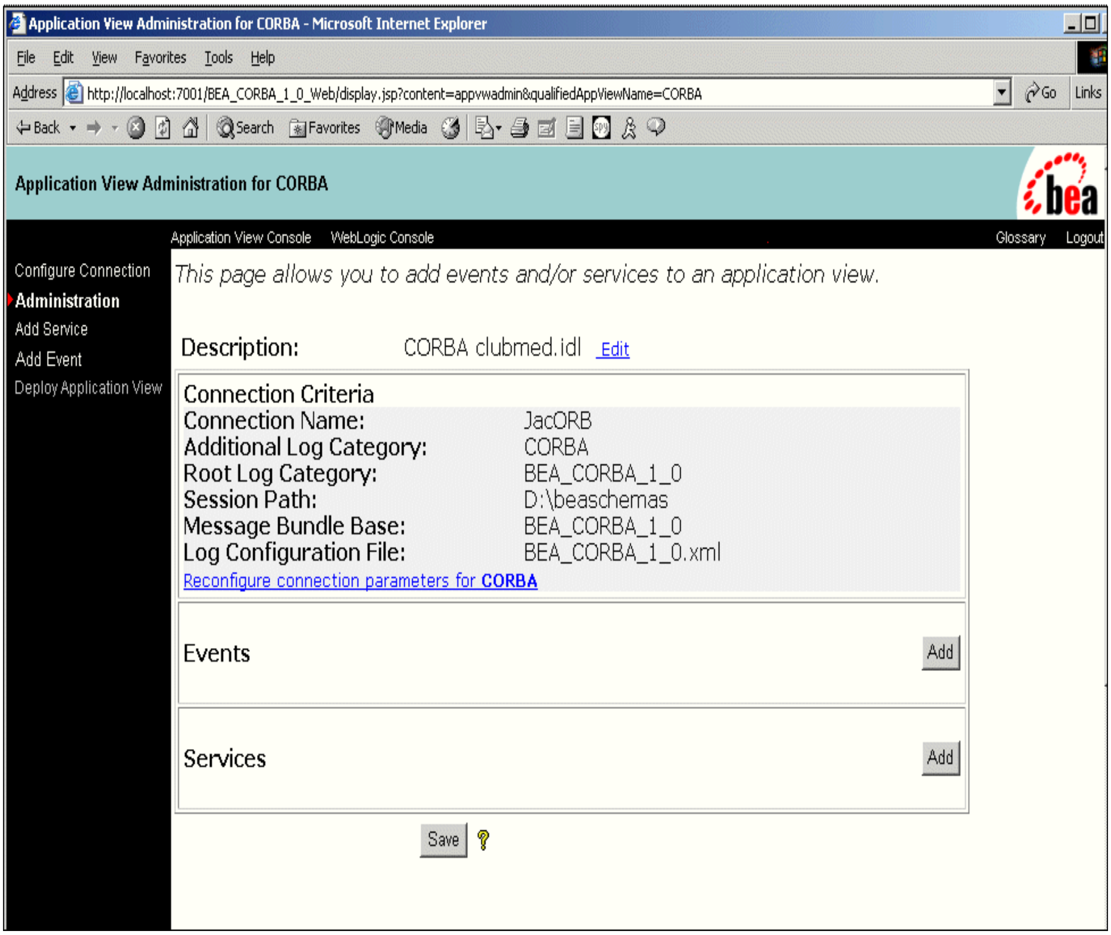
9. Enter the name of the BEA WebLogic Adapter for CORBA session path (sometimes known as the session base directory).

   This is the path that you specify when starting a BEA Application Explorer session, as described in "Connecting to an Object Request Broker" in Chapter 2, "Using the BEA Application Explorer With CORBA." The session path holds your ORB schema and connection information.

10. Select the session name—also known as the connection name—from the Connection name drop-down list.

11. Click Connect to EIS. The Application View Administration window opens.

Figure 3-6 Application View Administration Window


Application View Administration for CORBA - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:7001/BEA_CORBA_1_0_Web/display.jsp?content=appvwadmin&qualifiedAppViewName=CORBA
Back Search Favorites Media Go Links
Application View Administration for CORBA
Application View Console WebLogic Console Glossary Logout
Configure Connection
Administration
Add Service
Add Event
Deploy Application View
Add
Add
Save

This page allows you to add events and/or services to an application view.

Description: CORBA clubmed.idl Edit

Connection Criteria

Connection Name:	JacORB
Additional Log Category:	CORBA
Root Log Category:	BEA_CORBA_1_0
Session Path:	D:\beaschemas
Message Bundle Base:	BEA_CORBA_1_0
Log Configuration File:	BEA_CORBA_1_0.xml

Reconfigure connection parameters for CORBA

Events

Services

12. Click Save.

    You have finished creating the application view.

    You can now add services that support the application's functions, as described in "Adding a CORBA Service to an Application View" on page 3-10. Note that you must add a service before you deploy the new application view.

Adding a CORBA Service to an Application View

To add a service to an application view:

1. If it is not already open, open the application view to be modified:

   a. Log on to the WebLogic Application View Console at

      *//appserver-host:port/*wlai

      Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

   b. If prompted, enter a user name and password.

      Note: If the user name is not `system`, it must be included in the `adapter` group. For more information on adding the administrative server user name to the `adapter` group, see the *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*.

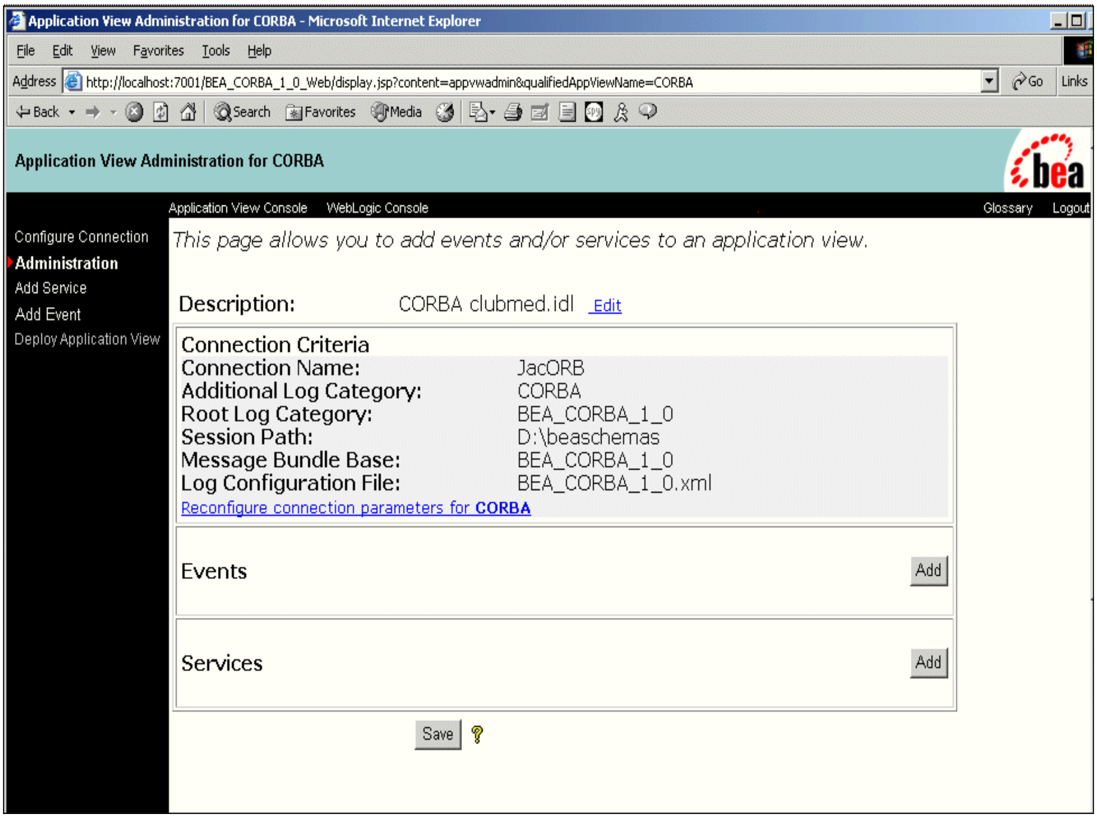
   c. Click Login.

      The WebLogic Integration Application View Console opens.

   d. Select the folder in which this application view resides, and then select the application view.

      The Administration window opens.

Figure 3-7 Application View Administration Window


Application View Administration for CORBA - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:7001/BEA_CORBA_1_0_Web/display.jsp?content=appvwadmin&qualifiedAppViewName=CORBA
Back Search Favorites Media
Application View Administration for CORBA
bea
Application View Console WebLogic Console Glossary Logout
Configure Connection
Administration
Add Service
Add Event
Deploy Application View
Add
Add
Save

This page allows you to add events and/or services to an application view.

Description: CORBA clubmed.idl Edit

Connection Criteria

Connection Name:	JacORB
Additional Log Category:	CORBA
Root Log Category:	BEA_CORBA_1_0
Session Path:	D:\beaschemas
Message Bundle Base:	BEA_CORBA_1_0
Log Configuration File:	BEA_CORBA_1_0.xml

Reconfigure connection parameters for **CORBA**

Events

Services

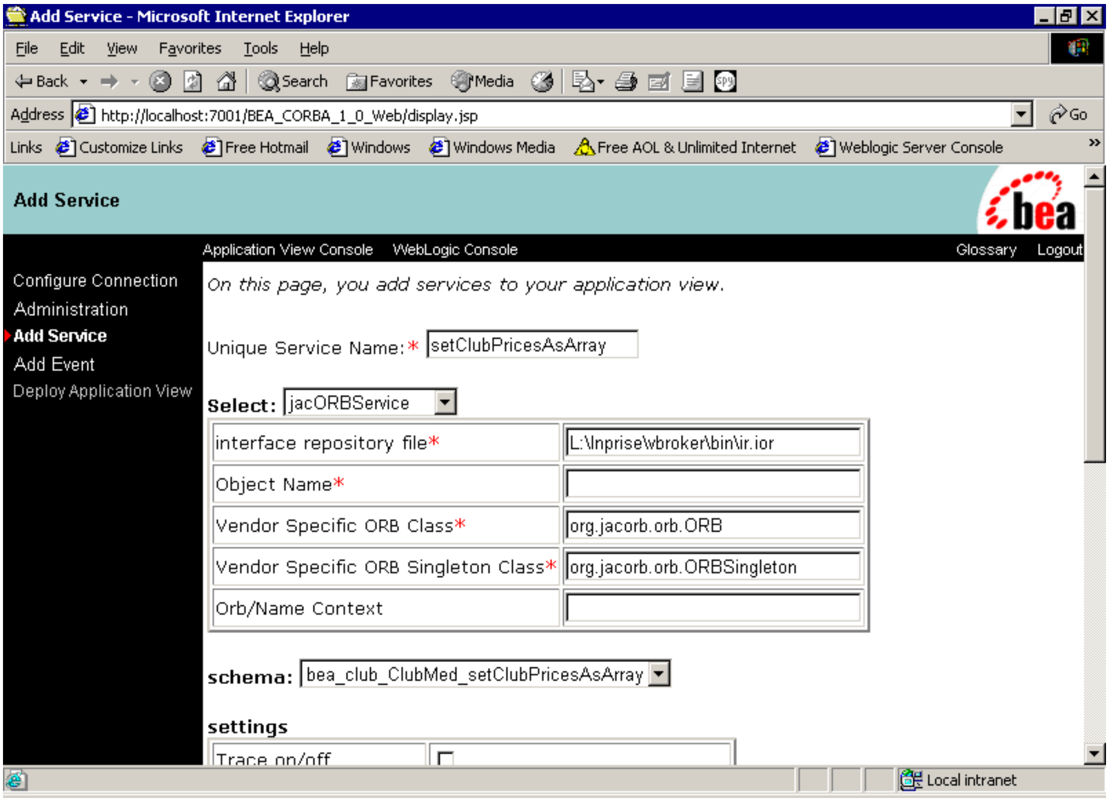
2. If the application view is deployed, you must undeploy it before adding the service. See “Optional Step: Undeploying an Application View” in “Defining an Application View” in *Using Application Integration*:

   - For WebLogic Integration 7.0, see `http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm`
   - For WebLogic Integration 2.1, see `http://edocs.bea.com/wlintegration/v2_1sp/aiuser/2usrdef.htm`

3. Click Add Service in the left pane.

   The Add Service window opens.

Figure 3-8 Add Service Window


Add Service - Microsoft Internet Explorer

Address http://localhost:7001/BEA_CORBA_1_0_Web/display.jsp

Add Service

On this page, you add services to your application view.

Unique Service Name:* setClubPricesAsArray

Select: jacORBService

interface repository file*	L:\Inprise\vbroker\bin\ir.ior
Object Name*	
Vendor Specific ORB Class*	org.jacorb.orb.ORB
Vendor Specific ORB Singleton Class*	org.jacorb.orb.ORBSingleton
Orb/Name Context	

schema: bea_club_ClubMed_setClubPricesAsArray

settings

Trace on/off ☐

4. Specify the service's properties, which are described in the following table. Required values are indicated with an asterisk (*).

Table 3-1 Service Properties

Property	Description
Unique Service Name	The name of the service that you are adding. The name must be unique within the application view. Valid characters include a-z, A-Z, 0-9, and underscore (_).

Table 3-1 Service Properties (Continued)

Property	**Description**
Select	The ORB to which the service request will be sent. The Application View Console automatically populates many of this window's fields with appropriate values based on the ORB that you choose.
interface repository file	The name and path of the interface repository file. The Application View Console automatically supplies this value based on the ORB that you chose.
Object Name	The name of the object (for example, `bea.clubmed`).
Vendor Specific ORB Class	The vendor-specific ORB class. The Application View Console automatically supplies this value based on the ORB that you chose.
Vendor Specific ORB Singleton Class	The vendor-specific ORB singleton class. The Application View Console automatically supplies this value based on the ORB that you chose.
Orb/Name Context	The ORB's name context.
Schema	The name of the schema that describes this service.
Trace on/off	Enables tracing for this service. Trace information is displayed in the runtime console.
Deep Debug on/off	Enables additional traces for deeper troubleshooting.
Logging on/off	Enables logging for this service. The log is written to `BEA_FILE_1_0.log` in the directory from which the application was started.
Maximum Log Size	Specify the maximum size of the log, in kilobytes.

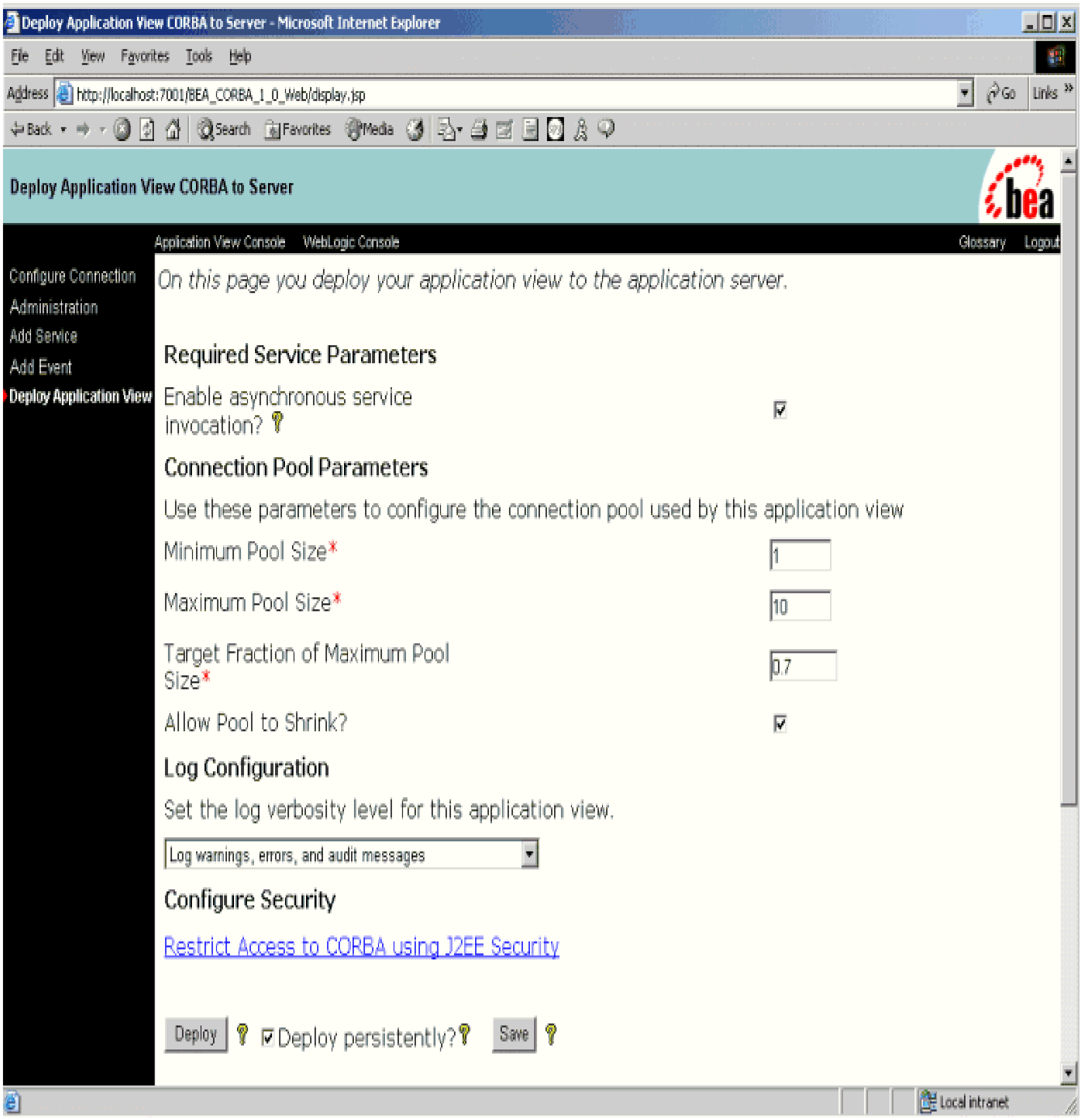
5. Click Add.

   The Administration window opens.

6. Click Continue.

   The Deploy Application View window opens.

Figure 3-9 Deploy Application View Window



Deploy Application View CORBA to Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:7001/BEA_CORBA_1_0_Web/display.jsp
Back Search Favorites Media
Application View Console WebLogic Console Glossary Logout
Configure Connection
Administration
Add Service
Add Event
Deploy Application View
Local intranet

Deploy Application View CORBA to Server

On this page you deploy your application view to the application server.

Required Service Parameters

Enable asynchronous service invocation? ☑

Connection Pool Parameters

Use these parameters to configure the connection pool used by this application view

Minimum Pool Size* 1

Maximum Pool Size* 10

Target Fraction of Maximum Pool Size* 0.7

Allow Pool to Shrink? ☑

Log Configuration

Set the log verbosity level for this application view.

Log warnings, errors, and audit messages

Configure Security

Restrict Access to CORBA using J2EE Security

Deploy ☑ Deploy persistently? Save

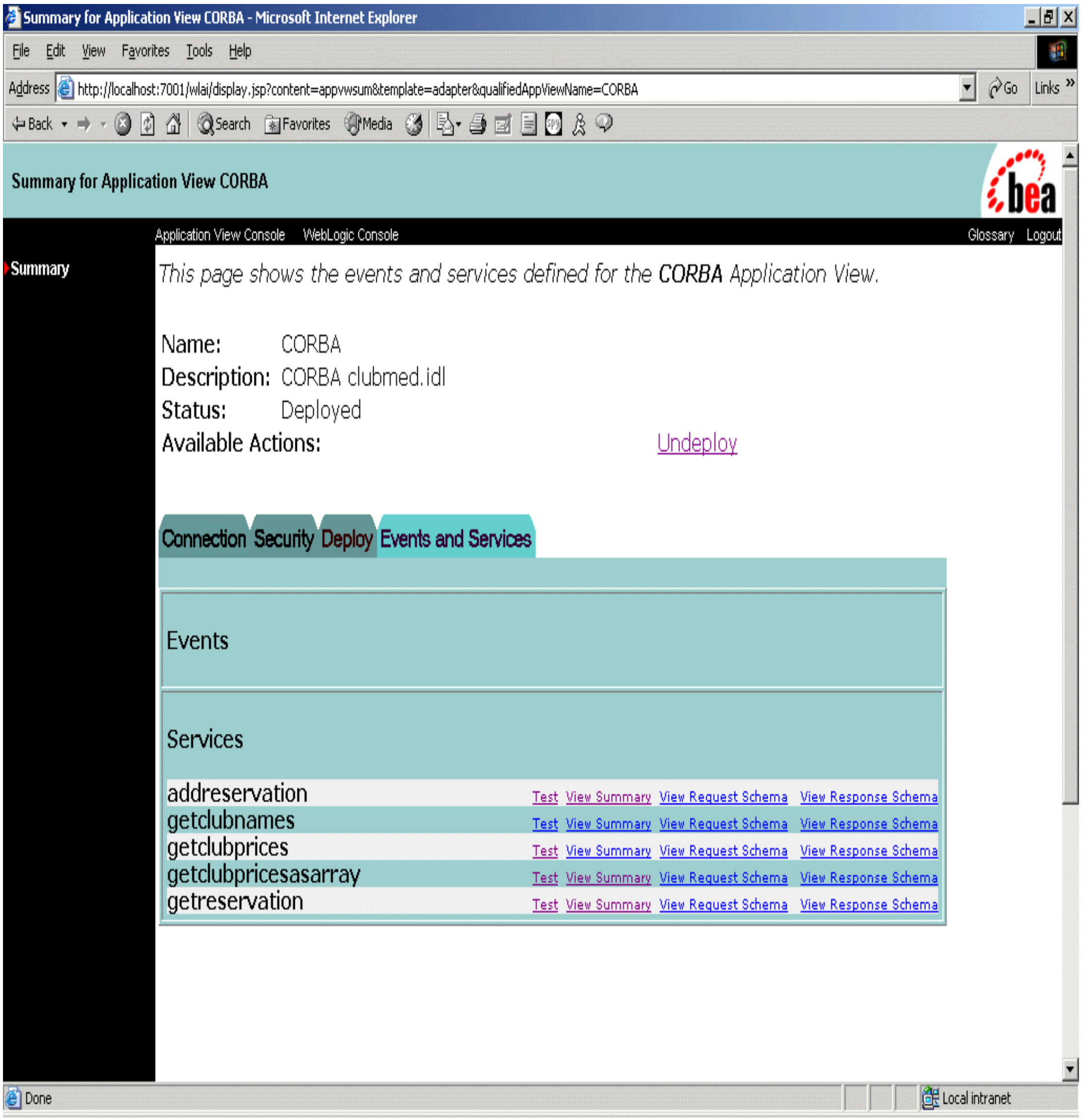
7. Update service parameters, connection pool parameters, log configuration, and security as necessary. For more information about these, see “Defining an Application View” in “Using Application Integration”:

   - For WebLogic Integration 7.0, see `http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm`
   - For WebLogic Integration 2.1, see `http://edocs.bea.com/wlintegration/v2_1sp/aiuser/2usrdef.htm`

8. Click Deploy to save and deploy the service.

   The Summary window for the application view opens.

Figure 3-10 Summary for Application View Window


Summary for Application View CORBA - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:7001/wlai/display.jsp?content=appvwsum&template=adapter&qualifiedAppViewName=CORBA
Back Search Favorites Media Go Links
Summary for Application View CORBA
Application View Console WebLogic Console Glossary Logout
Summary

This page shows the events and services defined for the CORBA Application View.

Name: CORBA
Description: CORBA clubmed.idl
Status: Deployed
Available Actions: Undeploy

Connection Security Deploy Events and Services

Events

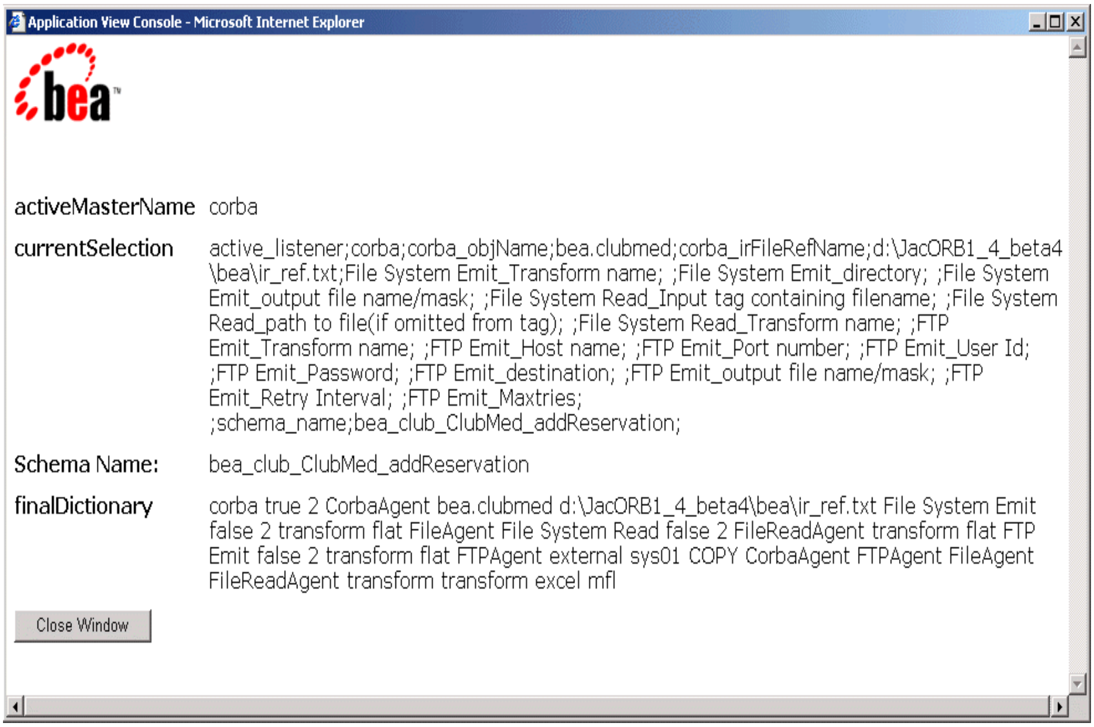
Services

addreservation	Test	View Summary	View Request Schema	View Response Schema
getclubnames	Test	View Summary	View Request Schema	View Response Schema
getclubprices	Test	View Summary	View Request Schema	View Response Schema
getclubpricesasarray	Test	View Summary	View Request Schema	View Response Schema
getreservation	Test	View Summary	View Request Schema	View Response Schema

9. To view a summary of the service as deployed, select the service and click View Summary.

A summary of the service is displayed in a new window.

Figure 3-11 Service Summary Window



activeMasterName	corba
currentSelection	active_listener;corba;corba_objName;bea.clubmed;corba_irFileRefName;d:\JacORB1_4_beta4 \bea\ir_ref.txt;File System Emit_Transform name; ;File System Emit_directory; ;File System Emit_output file name/mask; ;File System Read_Input tag containing filename; ;File System Read_path to file(if omitted from tag); ;File System Read_Transform name; ;FTP Emit_Transform name; ;FTP Emit_Host name; ;FTP Emit_Port number; ;FTP Emit_User Id; ;FTP Emit_Password; ;FTP Emit_destination; ;FTP Emit_output file name/mask; ;FTP Emit_Retry Interval; ;FTP Emit_Maxtries; ;schema_name;bea_club_ClubMed_addReservation;
Schema Name:	bea_club_ClubMed_addReservation
finalDictionary	corba true 2 CorbaAgent bea.clubmed d:\JacORB1_4_beta4\bea\ir_ref.txt File System Emit false 2 transform flat FileAgent File System Read false 2 FileReadAgent transform flat FTP Emit false 2 transform flat FTPAgent external sys01 COPY CorbaAgent FTPAgent FileAgent FileReadAgent transform transform excel mfl

You are now ready to test your application view, as described in "Testing an Application View" on page 18.

Testing an Application View

To test that an application view service interacts properly with the BEA WebLogic Adapter for CORBA:

1. If it is not already open, open the application view to be tested:

   a. Log on to the WebLogic Application View Console at

      `//appserver-host:port/wlai`

      Here, *`appserver-host`* is the IP address or host name where the WebLogic Integration Server is installed, and *`port`* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

   b. If prompted, enter a user name and password.

      Note: If the user name is not `system`, it must be included in the `adapter` group. For more information on adding the administrative server user name to the `adapter` group, see the *BEA WebLogic Adapter for CORBA Installation and Configuration Guide*.

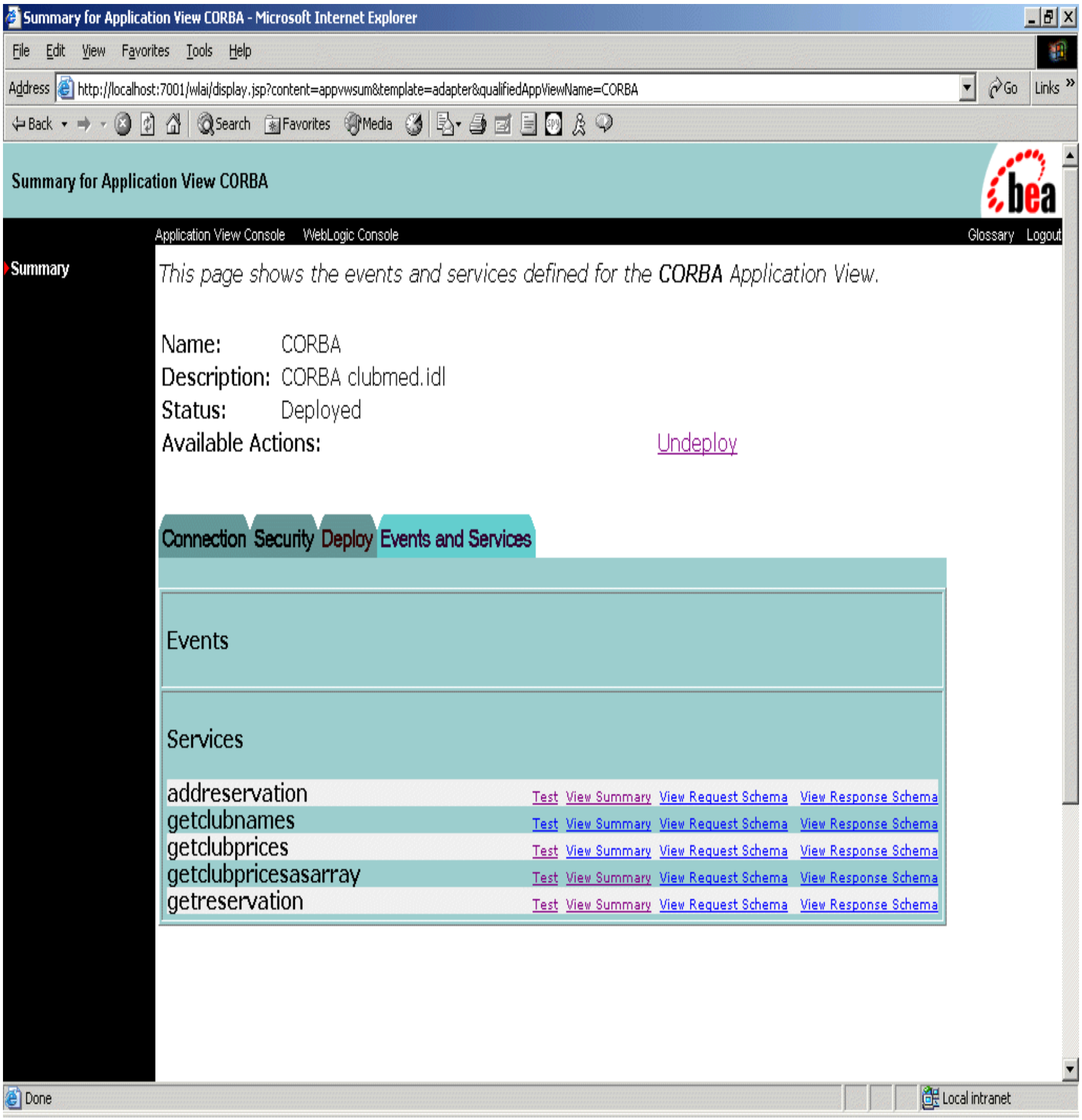
   c. Click Login.

      The WebLogic Integration Application View Console opens.

   d. Select the folder in which this application view resides, and then select the application view.

      The Summary window opens.

Figure 3-12 Displaying Services to Be Tested


Summary for Application View CORBA - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:7001/wlai/display.jsp?content=appvwsum&template=adapter&qualifiedAppViewName=CORBA
Go Links
Back Search Favorites Media
Application View Console WebLogic Console
Glossary Logout
Summary
Connection Security Deploy Events and Services
Done
Local intranet

Summary for Application View CORBA

This page shows the events and services defined for the CORBA *Application View.*

Name: CORBA
Description: CORBA clubmed.idl
Status: Deployed
Available Actions: Undeploy

Events

Services

addreservation	Test	View Summary	View Request Schema	View Response Schema
getclubnames	Test	View Summary	View Request Schema	View Response Schema
getclubprices	Test	View Summary	View Request Schema	View Response Schema
getclubpricesasarray	Test	View Summary	View Request Schema	View Response Schema
getreservation	Test	View Summary	View Request Schema	View Response Schema

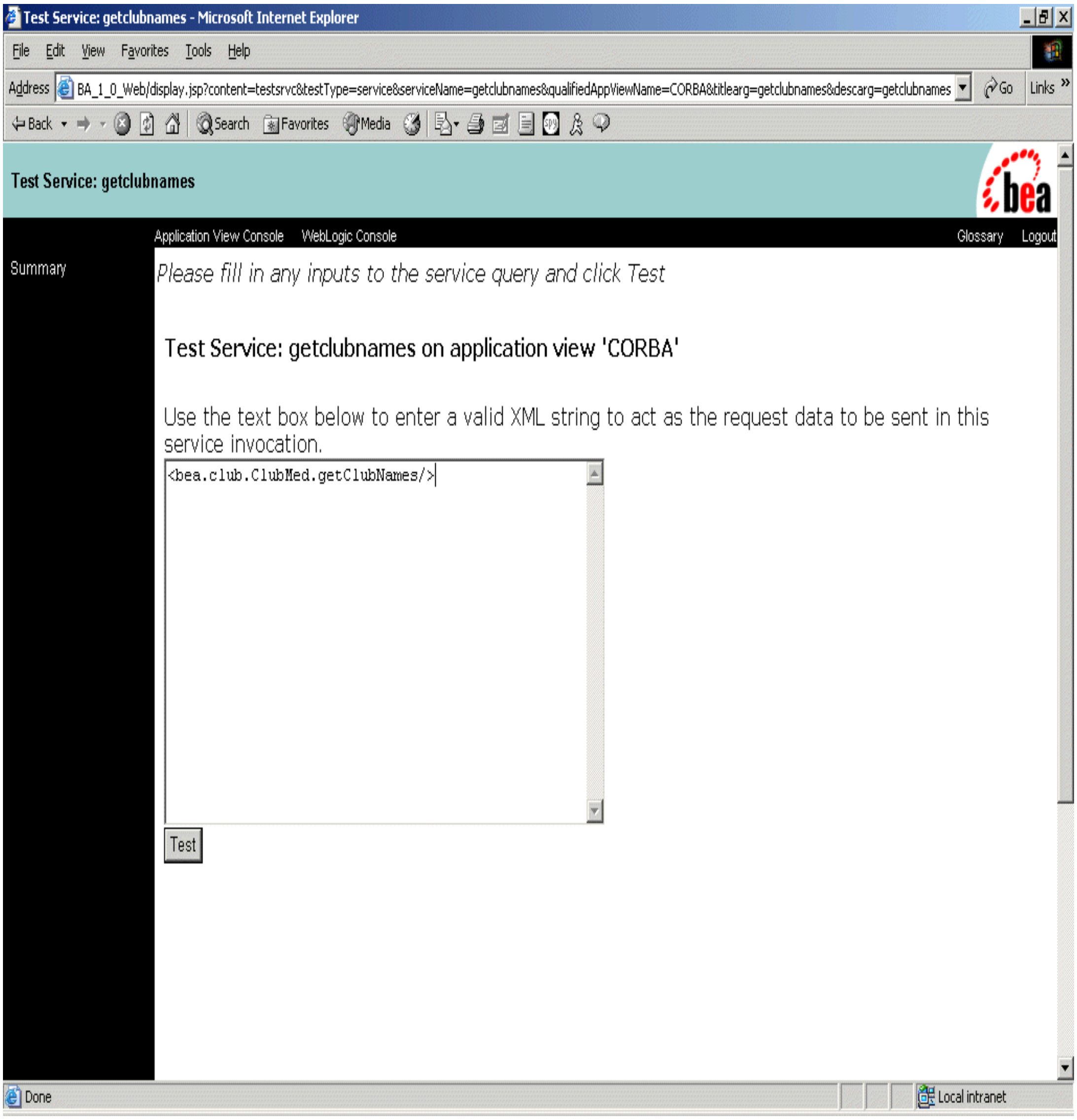
2. Select the service you wan to test and click Test.

   The Test Service window opens.

3. Enter a sample XML request document into the Test Service window, either by typing it or by copying and pasting it into the window. You can create your own request, or use one of the sample requests provided with the adapter and shown in Appendix B, “Sample Files.”

   The document should conform to the request schema of the service you are testing.

Figure 3-13 Entering a Request Into the Test Service Window


Test Service: getclubnames - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address BA_1_0_Web/display.jsp?content=testsrvc&testType=service&serviceName=getclubnames&qualifiedAppViewName=CORBA&titlearg=getclubnames&descarg=getclubnames
Go
Links
Back
Search
Favorites
Media
Test Service: getclubnames
Application View Console
WebLogic Console
Glossary
Logout
Summary
Test
Done
Local intranet

Please fill in any inputs to the service query and click Test

Test Service: getclubnames on application view 'CORBA'

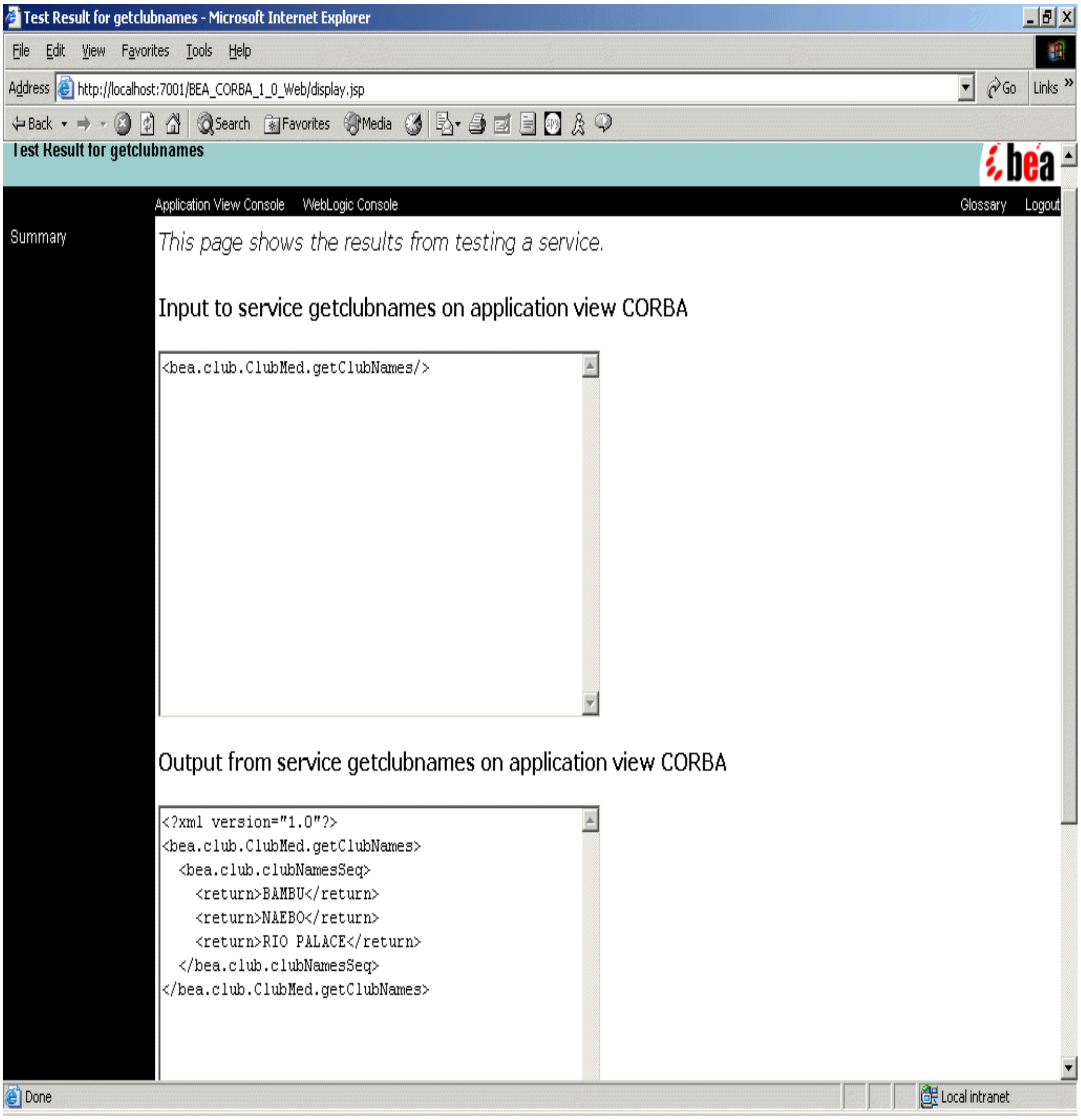
Use the text box below to enter a valid XML string to act as the request data to be sent in this service invocation.

```
<bea.club.ClubMed.getClubNames/>
```

4. Click Test.

If the test is successful, the Test Result window displays both the request and the response. The application view is successfully deployed.

Figure 3-14 Test Result Window


Test Result for getclubnames - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:7001/BEA_CORBA_1_0_Web/display.jsp
Go Links
Back Search Favorites Media
Test Result for getclubnames
bea
Application View Console WebLogic Console Glossary Logout
Summary
Done
Local intranet

This page shows the results from testing a service.

Input to service getclubnames on application view CORBA

```
<bea.club.ClubMed.getClubNames/>
```

Output from service getclubnames on application view CORBA

```
<?xml version="1.0"?>
<bea.club.ClubMed.getClubNames>
  <bea.club.clubNamesSeq>
    <return>BAMBU</return>
    <return>NAEBO</return>
    <return>RIO PALACE</return>
  </bea.club.clubNamesSeq>
</bea.club.ClubMed.getClubNames>
```

A Using CORBA Implementations With the Adapter

This section provides details about using the BEA WebLogic Adapter for CORBA with JacORB, Orbix2000, and VisiBroker for Java. It includes the following topics:

- Using JacORB With the BEA WebLogic Adapter for CORBA
- Using Orbix2000 With the BEA WebLogic Adapter for CORBA
- Using VisiBroker for Java With the BEA WebLogic Adapter for CORBA

Using JacORB With the BEA WebLogic Adapter for CORBA

The BEA WebLogic Adapter for CORBA includes a sample ORB called JacORB. JacORB is an open source Java implementation of the Object Management Group's CORBA specification. It is supplied with your software to enable you to test the adapter.

JacORB is designed to comply with CORBA 2.3 Java language mapping, and supports commonly used CORBA services. It runs on all platforms that implement the Java Virtual Machine (JVM). JacORB is made available under the terms of the GNU Library General Public License (LGPL). Commercial Support support is provided by Object Computing Inc., a Sun Authorized Java Center and member of the OMG; for more information, go to `http://www.ociweb.com`.

JacORB operates with any CORBA-compliant ORB over IIOP. In practice, JacORB has been used successfully with at least the following ORBs: MICO, TAO, Orbacus, Iona Orbix, Borland VisiBroker, ORBit, omniORB, Vitria C++ and Java. ORB interoperability is made simple by using a foreign name service IOR in the file where the BEA WebLogic Adapter for CORBA looks up the name server OAR. This can be configured in the `jacorb.properties` file.

The JacORB Name Service

Name servers are used to locate objects using a human–readable reference (a name) rather than a machine or network address. If objects providing a certain service are looked up using the service name, clients are separated from the actual locations of the objects that provide the service. The binding from name to service can be changed without the client's knowledge.

JacORB provides an implementation of the OMG's Interoperable Naming Service (INS), which supports the binding of names to object references (and looking up object references using these names). It also allows clients to easily convert names to strings and vice versa. The JacORB name service comprises two components: the name server program, and a set of interfaces and classes used to access the service.

The JacORB Interface Repository

Run–time type information in CORBA is managed by the ORB's Interface Repository (IFR) component. It allows applications to request, inspect, and modify IDL type information dynamically. For example, the IR enables applications to find out which operations an object supports. Some ORBs may also need the IR to find out whether a given object's type is a subtype of another, but most ORBs can operate without the IR by encoding this kind of type information in the helper classes generated by the IDL compiler.

In essence, the IR is just another remotely accessible CORBA object that offers operations to retrieve (and, theoretically, modify) type information. The IR manages type information in a hierarchical containment structure that corresponds to constructs within IDL specifications: modules contain definitions of interfaces, structures, constants etc. Interfaces in turn contain definitions of exceptions, operations, attributes and constants.

Building and Running the JacORB Request Broker

To build and run the JacORB Request Broker, perform the following steps. (You will need ANT 1.4.1, a Java-based build tool.)

1. Unzip `JacORB1_4_beta4-full.zip.`

   This file is found in BEA_CORBA_SAMPLES.zip in the adapter's installation directory. It creates the directory `JacORB1_4_beta4` on the selected drive. For example, if you unzip to drive D, the result is `D:\JacORB1_4_beta4`.

2. Unzip `beacorba.zip` in the directory `JacORB1_4_beta4` that you created in Step 1.

   This file is found in BEA_CORBA_SAMPLES.zip in the adapter's installation directory.

3. Copy `JacORB1_4_beta4\jacorb_properties.template` and rename the copy `JacORB1_4_beta4\jacorb.properties`.

4. Edit the `jacorb.properties` file:

   a. In the file's Initial References Configuration section, uncomment this line:

   ```
   #ORBInitRef.NameService=file:/d:/JacORB1_4_beta4/bea/ns_ref.txt
   ```

   b. In the same line, replace `d:` with the drive and path into which you unzipped your JacORB files.

   c. If the next line—`#ORBInitRef.NameService=file...`—is uncommented, comment it out.

5. Copy `JacORB1_4_beta4\bea\jaco.bat` into the `JacORB\bin` directory, replacing `JacORB1_4_beta4\bin\jaco.bat`.

6. Edit `JacORB1_4_beta4\bea\setenv-sample.bat` to specify directories in the following three statements:

   ```
   set JAVA_HOME=jdk_directory
   set JACORB_HOME=JacORB1_4_beta4_directory
   set ANT_HOME=ant_tool_directory
   ```

   Here, *`jdk_directory`* is the directory where your JDK resides, *`JacORB1_4_beta4_directory`* is the directory where JacORB resides, and *`ant_tool_directory`* is the directory where your ANT tool resides. For example:

   ```
   set JAVA_HOME=c:\jdk1.3
   set JACORB_HOME=d:\JacORB1_4_beta4
   set ANT_HOME=c:\jakarta-ant-1.4
   ```

7. Build the JacORB application. In a new DOS command window:

   a. Execute *`d:`*`JacORB1_4_beta4\bea\setenv-sample.bat`

   b. Execute *`d:`*`JacORB1_4_beta4\bea\club\ant`

   Here, *`d:`* is the drive and path into which you unzipped your JacORB files.

8. Start the Interface Repository service. In a new DOS command window:

   a. Execute `JacORB1_4_beta4\bea\setenv-sample.bat`

   b. Execute the following command:

   ```
   ir repository_class_path IOR_filename
   ```

   Here, *`repository_class_ path`* is the path to your repository class files, and *`IOR_filename`* is the name of the Interface Object Repository file. For example,

   ```
   JacORB1_4_beta4\bea\ir ..\classes ir_ref.txt
   ```

9. Start the name service. In a new DOS command window:

   a. Execute *`d:`*`\JacORB1_4_beta4\bea\setenv-sample.bat`

   Here, *`d:`* is the drive and path into which you unzipped your JacORB files.

   b. Execute the following command:

   ```
   ns [ins_filename][-p port] [-t timeout]
   ```

   Here, *`ins_filename`* is the name of the Naming Service file specified in the `jacorb.properties` file, *`port`* is the number of the port on which the service is listening, and *`timeout`* is the server timeout. For example,

   ```
   JacORB1_4_beta4\bea\ns ns_ref.txt
   ```

10. Start the Java interpreter explicitly by typing

    a. Execute *`d:`*`\JacORB1_4_beta4\bea\ setenv-sample.bat`

    Here, *`d:`* is the drive and path into which you unzipped your JacORB files.

    b. Execute the following command:

    ```
    jaco jacorb.naming.NameServer [filename][-p port][-t timeout]
    ```

    Here, *`jacorb.naming.NameServer`* is the name of the Name Server, *`filename`* is the name of the Naming Service file specified in the `jacorb.properties` file, *`port`* is the number of the port on which the service is listening, and *`timeout`* is the server timeout. For example,

    ```
    JacORB1_4_beta4\bea\jaco bea.club.ClubServer
    ```

Using Orbix2000 With the BEA WebLogic Adapter for CORBA

The BEA WebLogic Adapter for CORBA supports Orbix 2000 Versions 1.2 and communicates using IIOP version 1.1.

Orbix is a software environment for building and integrating distributed object-oriented applications. Orbix is a full implementation of the Common Object Request Broker Architecture (CORBA) from the Object Management Group (OMG). Orbix fully supports CORBA version 2.3. For more information, see the *Orbix 2000 Administrator's Guide*.

Orbix includes a CORBA IDL compiler, which is used by programmers to compile interface definitions along with the client and server code. A client application compiled in this way contains internal information about server objects. Clients use this information to invoke the remote objects. Orbix provides an interface repository, which enables clients to call operations on Interface Definition Language (IDL) interfaces that are unknown at compile time. The interface repository (IFR) provides centralized persistent storage of IDL interfaces. Orbix programs can query the interface repository at runtime to obtain information about IDL definitions.

To verify that your Orbix server is registered in the Naming Service, enter the following command at the command prompt:

```
itadmin ns list
```

Your server name should appear within a list of all the registered servers. If it is not in the list, you must register your server.

To ensure your IFR is running and populated, you can view its contents using the following command:

```
itadmin ifr list
```

This command lists all the currently scoped names, such as interfaces and types.

You can use the itadmin command to view the IDL definition of one of the current scoped names, as follows:

```
itadmin ifr show current_scoped_name
```

In order to populate the interface repository with IDL definitions, run the IDL compiler with the `-R` option. For example, the following command populates the interface repository with the IDL definitions in `bank.idl`:

```
idl -I. -I$(ART_IDL_DIR)\omg -R= bank.idl
```

For more information on the Orbix utilities and command line, see Section IV of the *Orbix 2000 Administrators Guide.*

Using VisiBroker for Java With the BEA WebLogic Adapter for CORBA

VisiBroker is a complete CORBA 2.3 Object Request Broker (ORB) that supports the development, deployment, and management of distributed object applications across a variety of hardware platforms and operating systems. In addition to VisiBroker (the ORB), three other components are available with VisiBroker:

- Naming Service, which allows you to associate one or more logical names with an object implementation and to store those names in a namespace. It also lets client applications use this service to obtain an object reference using the logical name assigned to that object.
- Event Service, which provides a facility that separates the communication between objects. It provides a *supplier-consumer* communications model that allows multiple *supplier objects* to send data asynchronously to multiple *consumer objects* through an event channel.
- Gatekeeper, which runs on a Web server and enables client programs to locate and use objects that do not reside on the Web server and to receive callbacks, even when firewalls are being used. The Gatekeeper can also be used as an HTTP daemon, thereby eliminating the requirement for a separate HTTP server during the application development phase.

The BEA WebLogic Adapter for CORBA supports VisiBroker for Java Version 4.5 and communicates using IIOP version 1.1. Applications created with VisiBroker for Java can communicate with object implementations developed with VisiBroker for C++

VisiBroker requires the Java Development Kit (JDK) or the Java Runtime Environment (JRE). You can obtain these tools from the Sun Microsystems Web site (`http://java.sun.com/`). JRE version 1.2.2 or higher is required to run the VisiBroker Console. You must install the JRE before you install VisiBroker. However, VisiBroker supports any current version of Java for your applications.

The BEA WebLogic Adapter for CORBA requires that an IOR file be available to locate the reference to the Interface Repository. However, VisiBroker's IOR file is not automatically output to a file. Modify your IR startup procedure to automatically output the startup IOR reference to a file. For example, use the following command to automatically output the IOR reference to `ir.ior`:

```
irep myIr >ir.ior
```

VisiBroker configuration and run-time requirements for using the BEA WebLogic Adapter for CORBA include:

- The PATH environment variable must point to the VisiBroker libraries used by BEA WebLogic Adapter for CORBA.
- The VisiBroker jar file must be in the class path.
- The IFR must be populated with the IDL of the objects for which you want to create Web services.
- The VisiBroker Naming Service and IFR must be running.
- The CORBA servers you are going to use must be running or be set up to start on demand.

For example, with VisiBroker you start the Interface Repository using the following command:

```
irep myIr >ir.ior
```

Here, *`myIR`* is the startup IOR reference.

With VisiBroker you start the naming service using the following command:

```
$start nameserv NS_name
```

Here, *`NS_name`* is the name of the naming service.

To verify that your server is registered in the Naming Service and your IFR is loaded:

1. Choose Start→Programs→VisiBroker→VisiBroker Console.
2. Expand VisiBroker ORB Services.
3. Expand the Naming Services folder or the IFR folder.

   A list of the Naming Service or IFR objects should appear in the right pane.

Alternately, you can enter the following command at a command prompt:

```
osfind
```

This command finds the name of the server running the VisiBroker Naming Service. It is usually the machine where VisiBroker is installed.

For more information, refer to *VisiBroker for Java Installation Guide*, available at

`http://info.borland.com/techpubs/books/vbj/vbj45/framesetindex.html`

B Sample Files

This section describe the sample files delivered with the software. It includes the following topics:

- Sample XML Request and Response Documents
- Definitions for ClubMed Object

Sample XML Request and Response Documents

This section includes the following sample request and response documents:

- addGetReservation Request
- addGetReservation Responses
- AddReservation Request
- AddReservation Response
- addReservationComplex Request
- addReservationComplex Response
- addReservationWithOut Request
- addReservationWithOut Response
- cancelReservation Request

- cancelReservation Response
- GetClubNames Request
- GetClubNames Response
- getClubNames4 Request
- getClubNames4 Response
- GetClubPrices Request
- GetClubPrices Response
- GetClubPricesAsArray Request
- GetClubPricesAsArray Response
- GetReservation Request
- GetReservation Response
- getReservationAsOut Request
- getReservationAsOut Response
- setCancel Request
- setCancel Response
- setClubPrices Request
- setClubPrices Response
- setClubPricesAsArray Request
- setClubPricesAsArray Response
- setStatus Request
- setStatus Response
- status Request
- status Response
- testOneDim Request
- testOneDim Response

- testTwoDim Request
- testTwoDim Response
- testTypes Request
- testTypes Response

addGetReservation Request

Listing B-1 addGetReservation Request

```
<bea.club.ClubMed.addGetReservation>
  <club>BAMBU</club>
  <resvData>
    <fname>Bill</fname>;
    <lname>Wales</lname>
    <weekDate>Monday</weekDate>
    <street>Big Bungalow</street>
    <city>High End</city>
    <state>WA</state>
    <zip>99990</zip>
    <phone>123-123-3456</phone>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
    <resv>1000</resv>
  </resvData>
</bea.club.ClubMed.addGetReservation>
```

addGetReservation Response

Listing B-2 addGetReservation Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <bea.club.ClubMed.addGetReservation>
  <return>1002</return>
    <resvData>
    <fname>Bill</fname>
    <lname>Wales</lname>
    <weekDate>Monday</weekDate>
    <street>Big Bungalow</street>
    <city>High End</city>
    <state>WA</state>
    <zip>99990</zip>
    <phone>123-123-3456</phone>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
    <resv>1000</resv>
  </resvData>
</bea.club.ClubMed.addGetReservation>
```

AddReservation Request

Listing B-3 AddReservation Request

```
<bea.club.ClubMed.addReservation>
  <club>BAMBU</club>
  <resvData>
    <fname>Bill</fname>;
    <lname>Wales</lname>
    <weekDate>Monday</weekDate>
    <address>Big Bungalow</address>
    <city>High End</city>
    <state>WA</state>
    <zip>99990</zip>
    <phone>123-123-3456</phone>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
  </resvData>
</bea.club.ClubMed.addReservation>
```

AddReservation Response

Listing B-4 AddReservation Response

```
<?xml version="1.0"?>
<bea.club.ClubMed.addReservation>
  <return>1001</return>
</bea.club.ClubMed.addReservation>
```

addReservationComplex Request

Listing B-5 addReservationComplex Request

```
<bea.club.ClubMed.addReservationComplex>
  <club>BAMBU</club>
  <resvData>
     <addr>
        <fname>Bill</fname>;
        <lname>Wales</lname>
        <street>Big Bungalow</street>
        <city>High End</city>
        <state>WA</state>
        <zip>99990</zip>
        <phone>123-123-3456</phone>
        <prices>
          <bea.club.pricesStruct>
            <date>2002-11-23</date>
            <adultFare>88</adultFare>
            <childFare>50</childFare>
          </bea.club.pricesStruct>
          <bea.club.pricesStruct>
            <date>2002-11-24</date>
            <adultFare>88</adultFare>
            <childFare>50</childFare>
          </bea.club.pricesStruct>
          <bea.club.pricesStruct>
            <date>2002-11-25</date>
            <adultFare>88</adultFare>
            <childFare>50</childFare>
          </bea.club.pricesStruct>
          <bea.club.pricesStruct>
            <date>2002-11-26</date>
            <adultFare>88</adultFare>
            <childFare>50</childFare>
          </bea.club.pricesStruct>
        </prices>
      </addr>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
    <clubs>
      <item>BAMBU</item>
      <item>RIU PALACE</item>
    </clubs>
  </resvData>
</bea.club.ClubMed.addReservationComplex>
```

addReservationComplex Response

Listing B-6 addReservationComplex Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.addReservationComplex>
  <return>1000</return>
  <resvData>
    <addr>
      <fname>Bill</fname>
      <lname>Wales</lname>
      <street>Big Bungalow</street>
      <city>High End</city>
      <state>WA</state>
      <zip>99990</zip>
      <phone>123-123-3456</phone>
      <prices>
        <bea.club.pricesArray>
          <date>2002-11-23</date>
          <adultFare>88</adultFare>
          <childFare>50</childFare>
        </bea.club.pricesArray>
        <bea.club.pricesArray>
          <date>2002-11-24</date>
          <adultFare>88</adultFare>
          <childFare>50</childFare>
        </bea.club.pricesArray>
        <bea.club.pricesArray>
          <date>2002-11-25</date>
          <adultFare>88</adultFare>
          <childFare>50</childFare>
        </bea.club.pricesArray>
        <bea.club.pricesArray>
          <date>2002-11-26</date>
          <adultFare>88</adultFare>
          <childFare>50</childFare>
        </bea.club.pricesArray>
      </prices>
    </addr>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
    <clubs>
      <item>BAMBU</item>
      <item>RIU PALACE</item>
    </clubs>
  </resvData>
</bea.club.ClubMed.addReservationComplex>
```

addReservationWithOut Request

Listing B-7 addReservationWithOut Request

```
<bea.club.ClubMed.addReservationWithOut>
  <club>BAMBU</club>
  <resvData>
    <fname>Bill</fname>;
    <lname>Wales</lname>
    <weekDate>Monday</weekDate>
    <street>Big Bungalow</street>
    <city>High End</city>
    <state>WA</state>
    <zip>99990</zip>
    <phone>123-123-3456</phone>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
    <resv>0</resv>
  </resvData>
</bea.club.ClubMed.addReservationWithOut>
```

addReservationWithOut Response

Listing B-8 addReservationWithOut Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.addReservationWithOut>
  <num>1004</num>
</bea.club.ClubMed.addReservationWithOut>
```

cancelReservation Request

Listing B-9 cancelReservation Request

```
<bea.club.ClubMed.cancelReservation>
  <resv>1000</resv>
</bea.club.ClubMed.cancelReservation>
```

cancelReservation Response

Listing B-10 cancelReservation Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.cancelReservation>
  <return>false</return>
</bea.club.ClubMed.cancelReservation>
```

GetClubNames Request

Listing B-11 GetClubNames Request

```
<bea.club.ClubMed.getClubNames>
</bea.club.ClubMed.getClubNames>
```

GetClubNames Response

Listing B-12 GetClubNames Response

```
<?xml version="1.0"?>
<bea.club.ClubMed.getClubNames>
  <bea.club.clubNamesSeq>
    <return>BAMBU</return>
    <return>NAEBO</return>
    <return>RIO PALACE</return>
  </bea.club.clubNamesSeq>
</bea.club.ClubMed.getClubNames>
```

getClubNames4 Request

Listing B-13 getClubNames4 Request

```
<bea.club.ClubMed.getClubNames4>
</bea.club.ClubMed.getClubNames4>
```

getClubNames4 Response

Listing B-14 getClubNames4 Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.getClubNames4>
  <return>
    <item>BAMBU</item>
    <item>NAEBO</item>
    <item>RIU PALACE</item>
  </return>
</bea.club.ClubMed.getClubNames4>
```

GetClubPrices Request

Listing B-15 GetClubPrices Request

```
<bea.club.ClubMed.getClubPrices>
  <club>BAMBU</club>
</bea.club.ClubMed.getClubPrices>
```

GetClubPrices Response

Listing B-16 GetClubPrices Response

```
<?xml version="1.0"?>
<bea.club.ClubMed.getClubPrices>
  <bea.club.pricesSeq>
    <bea.club.pricesStruct>
      <date>2002-11-23</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-24</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-25</date>
      <adultFare>80</adultFare>
      <childFare>45</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-26</date>
      <adultFare>80</adultFare>
      <childFare>45</childFare>
    </bea.club.pricesStruct>
  </bea.club.pricesSeq>
</bea.club.ClubMed.getClubPrices>
```

GetClubPricesAsArray Request

Listing B-17 GetClubPricesAsArray Request

```
<bea.club.ClubMed.getClubPricesAsArray>
  <club>BAMBU</club>
</bea.club.ClubMed.getClubPricesAsArray>
```

GetClubPricesAsArray Response

Listing B-18 GetClubPricesAsArray Response

```
<?xml version="1.0"?>
<bea.club.ClubMed.getClubPricesAsArray>
  <bea.club.pricesArray>
    <bea.club.pricesStruct>
      <date>2002-11-23</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-24</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-25</date>
      <adultFare>80</adultFare>
      <childFare>45</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-26</date>
      <adultFare>80</adultFare>
      <childFare>45</childFare>
    </bea.club.pricesStruct>
  </bea.club.pricesArray>
</bea.club.ClubMed.getClubPricesAsArray>
```

GetReservation Request

Listing B-19 GetReservation Request

```
<bea.club.ClubMed.getReservation>
  <resv>1001</resv>
</bea.club.ClubMed.getReservation>
```

GetReservation Response

Listing B-20 GetReservation Response

```
<?xml version="1.0"?>
<bea.club.ClubMed.getReservation>
  <bea.club.resvStruct>
    <fname>Bill</fname>
    <lname>Wales</lname>
    <weekDate>Monday</weekDate>
    <address>Big Bungalow</address>
    <city>High End</city>
    <state>WA</state>
    <zip>99990</zip>
    <phone>123-123-3456</phone>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
  </bea.club.resvStruct>
</bea.club.ClubMed.getReservation>
```

getReservationAsOut Request

Listing B-21 getReservationAsOut Request

```
<bea.club.ClubMed.getReservationAsOut>
  <resv>1000</resv>
</bea.club.ClubMed.getReservationAsOut>
```

getReservationAsOut Response

Listing B-22 getReservationAsOut Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.getReservationAsOut>
  <resvData>
    <addr>
      <fname>Bill</fname>
      <lname>Wales</lname>
      <street>Big Bungalow</street>
      <city>High End</city>
      <state>WA</state>
      <zip>99990</zip>
      <phone>123-123-3456</phone>
      <prices>
        <bea.club.pricesArray>
          <date>2002-11-23</date>
          <adultFare>88</adultFare>
          <childFare>50</childFare>
        </bea.club.pricesArray>
        <bea.club.pricesArray>
          <date>2002-11-24</date>
          <adultFare>88</adultFare>
          <childFare>50</childFare>
        </bea.club.pricesArray>
        <bea.club.pricesArray>
          <date>2002-11-25</date>
          <adultFare>88</adultFare>
          <childFare>50</childFare>
        </bea.club.pricesArray>
        <bea.club.pricesArray>
          <date>2002-11-26</date>
          <adultFare>88</adultFare>
          <childFare>50</childFare>
        </bea.club.pricesArray>
      </prices>
    </addr>
    <totalFare>5000.0</totalFare>
    <partyAdults>2</partyAdults>
    <partyChildren>2</partyChildren>
    <date>2001-09-21</date>
    <clubs>
      <item>BAMBU</item>
      <item>RIU PALACE</item>
    </clubs>
  </resvData>
</bea.club.ClubMed.getReservationAsOut>
```

setCancel Request

Listing B-23 setCancel Request

```
<bea.club.ClubMed.setCancel>
  <resv>1000</resv>
  <cancel>false</cancel>
</bea.club.ClubMed.setCancel>
```

setCancel Response

Listing B-24 setCancel Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.setCancel/>
```

setClubPrices Request

Listing B-25 setClubPrices Request

```
<bea.club.ClubMed.setClubPrices>
  <club>BAMBU</club>
  <prices>
    <bea.club.pricesStruct>
      <date>2002-11-23</date>
      <adultFare>88</adultFare>

      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-24</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
  </prices>
</bea.club.ClubMed.setClubPrices>
```

setClubPrices Response

Listing B-26 setClubPrices Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.setClubPrices/>
```

setClubPricesAsArray Request

Listing B-27 setClubPricesAsArray Request

```
<bea.club.ClubMed.setClubPricesAsArray>
  <club>BAMBU</club>
  <prices>
    <bea.club.pricesStruct>
      <date>2002-11-23</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-24</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-24</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
    <bea.club.pricesStruct>
      <date>2002-11-24</date>
      <adultFare>88</adultFare>
      <childFare>50</childFare>
    </bea.club.pricesStruct>
  </prices>
</bea.club.ClubMed.setClubPricesAsArray>
```

setClubPricesAsArray Response

Listing B-28 setClubPricesAsArray Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.setClubPricesAsArray/>
```

setStatus Request

Listing B-29 setStatus Request

```
<bea.club.ClubMed.setStatus>
  <resv>1000</resv>
  <cod>CANCELED</cod>
</bea.club.ClubMed.setStatus>
```

setStatus Response

Listing B-30 setStatus Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.setStatus/>
```

status Request

Listing B-31 status Request

```
<bea.club.ClubMed.status>
  <resv>1000</resv>
</bea.club.ClubMed.status>
```

status Response

Listing B-32 status Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.status>
  <return>PROCESSED</return>
</bea.club.ClubMed.status>
```

testOneDim Request

Listing B-33 testOneDim Request

```
<bea.club.ClubMed.testOneDim>
  <par>
    <item>1</item>
    <item>2</item>
    <item>3</item>
  </par>
</bea.club.ClubMed.testOneDim>
```

testOneDim Response

Listing B-34 testOneDim Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.testOneDim>
  <par>
    <item>1</item>
    <item>2</item>
    <item>3</item>
  </par>
</bea.club.ClubMed.testOneDim>
```

testTwoDim Request

Listing B-35 testTwoDim Request

```
<bea.club.ClubMed.testTwoDim>
  <par>
    <bea.club.twoDim_row>
      <item>one</item>
      <item>two</item>
      <item>three</item>
      <item>four</item>
      <item>five</item>
    </bea.club.twoDim_row>
    <bea.club.twoDim_row>
      <item>one</item>
      <item>two</item>
      <item>three</item>
      <item>four</item>
      <item>five</item>
    </bea.club.twoDim_row>
    <bea.club.twoDim_row>
      <item>one</item>
      <item>two</item>
      <item>three</item>
      <item>four</item>
      <item>five</item>
    </bea.club.twoDim_row>
  </par>
</bea.club.ClubMed.testTwoDim>
```

testTwoDim Response

Listing B-36 testTwoDim Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.testTwoDim>
  <par>
    <bea.club.twoDim_row>
      <item>one</item>
      <item>two</item>
      <item>three</item>
      <item>four</item>
      <item>five</item>
    </bea.club.twoDim_row>
    <bea.club.twoDim_row>
      <item>one</item>
      <item>two</item>
      <item>three</item>
      <item>four</item>
      <item>five</item>
    </bea.club.twoDim_row>
    <bea.club.twoDim_row>
      <item>one</item>
      <item>two</item>
      <item>three</item>
      <item>four</item>
      <item>five</item>
    </bea.club.twoDim_row>
  </par>
</bea.club.ClubMed.testTwoDim>
```

testTypes Request

Listing B-37 testTypes Request

```
<bea.club.ClubMed.testTypes>
  <all>
    <t1>string10</t1>
    <t2>1000</t2>
    <t3>1000999</t3>
    <t4>12345</t4>
    <t5>99999999</t5>
    <t6>1.234</t6>
    <t7>98765.1234</t7>
    <t8>c</t8>
    <t9>false</t9>
    <t10>34</t10>
    <t11>w</t11>
    <t14>123456789012899</t14>
    <t15>12345678901289999</t15>
  </all>
</bea.club.ClubMed.testTypes>
```

testTypes Response

Listing B-38   testTypes Response

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bea.club.ClubMed.testTypes>
  <all>
    <t1>string10</t1>
    <t2>1000</t2>
    <t3>1000999</t3>
    <t4>12345</t4>
    <t5>99999999</t5>
    <t6>1.234</t6>
    <t7>98765.1234</t7>
    <t8>c</t8>
    <t9>false</t9>
    <t10>34</t10>
    <t11>w</t11>
    <t14>123456789012899</t14>
    <t15>12345678901289999</t15>
  </all>
</bea.club.ClubMed.testTypes>
```

Definitions for ClubMed Object

The BEA WebLogic Adapter for CORBA provides the sample ClubMed object to enable you to test the adapter.

JacORB.properties File

The following listing reflects the JacORB.properties file after you have edited it according to the instructions in "Building and Running the JacORB Request Broker" in "Using JacORB With the BEA WebLogic Adapter for CORBA" in Appendix A, "Using CORBA Implementations With the Adapter."

Listing B-39 JacORB.properties File

```
##
##  JacORB configuration options
##

########################################
#                                      #
#   Initial references configuration   #
#                                      #
########################################

#
# URLs where IORs are stored (used in orb.resolve_initial_service())
# DO EDIT these! (Only those that you are planning to use,
# of course ;-).
#
# The ORBInitRef references are created on ORB startup time. In the
# cases of the services themselves, this may lead to exceptions being
# displayed (because the services aren't up yet). These exceptions
# are handled properly and cause no harm!

#ORBInitRef.NameService=corbaloc::160.45.110.41:38693/StandardNS/NameServer%2DP
OA/_root
ORBInitRef.NameService=file:/d:/JacORB1_4_beta4/bea/ns_ref.txt
#ORBInitRef.NameService=http://www.x.y.z/~user/NS_Ref
#ORBInitRef.TradingService=http://www.x.y.z/~user/TraderRef
```

```
# JacORB-specific URLs
jacorb.ProxyServerURL=http://www.x.y.z/~user/Appligator_Ref


##################################
#                                #
#   ORB version number output    #
#                                #
##################################

# if on, the ORB's version number is printed
# any time the ORB is initialized
jacorb.orb.print_version=on

##################################
#                                #
#   Debug output configuration   #
#                                #
##################################

# use (java) jacorb.util.CAD to generate an apropriate
# verbosity level
# 0 = off
# 1 = important messages and exceptions
# 2 = informational messages and exceptions
# >= 3 = debug-level output (may confuse the unaware user :-)
jacorb.verbosity=1

# where does output go? Terminal is default
#jacorb.logfile=LOGFILEPATH

# hexdump outgoing messages
jacorb.debug.dump_outgoing_messages=off

# hexdump incoming messages
jacorb.debug.dump_incoming_messages=off


###################################################
#                                                 #
#    WARNING: The following properties should    #
#    only be edited by the expert user. They     #
#    can be left untouched in most cases!        #
#                                                 #
###################################################
```

```
################################
#                              #
#   Basic ORB Configuration    #
#                              #
################################

# the GIOP minor version number to use for newly created IORs
jacorb.giop_minor_version=2

# number of retries if connection cannot directly be established
jacorb.retries=5

# how many msecs. do we wait between retries
jacorb.retry_interval=500

# size of network buffers for outgoing messages
jacorb.outbuf_size=2048

# log2 of maximum buffer size managed by the internal
# buffer manager.
#
# This is NOT the maximum buffer size that
# can be used, but just the largest size of buffers that
# will be kept and managed. This value will be added to
# an internal constant of 5, so the real value in bytes
# is 2**(5+maxManagedBufSize-1). You only need to increase this
# value if you are dealing with LOTS of LARGE data structures.
# You may decrease it to make the buffer manager release large
# buffers immediately rather than keeping them for later
# reuse.
jacorb.maxManagedBufSize=18

# client-side timeout, set no non-zero to stop blocking
# after so many msecs.
#jacorb.connection.client_timeout=0

# max time a server keeps a connection open if nothing happens
#jacorb.connection.server_timeout=10000

#jacorb.reference_caching=off

#
# The following property specifies the class which is used for
# reference caching. WeakHashtable uses WeakReferences, so entries
# get gc'ed if only the Hashtable has a reference to them. This
# is useful if you have many references to short-living non-persistent
# CORBA objects. It is only available for java 1.2 and above.
#
# On the other hand the standard Hashtable keeps the references until
# they are explicitely deleted by calling _release(). This is useful
# for persistent and long-living CORBA objects.
#
#jacorb.hashtable_class=org.jacorb.util.WeakHashtable
#
jacorb.hashtable_class=java.util.Hashtable

# use GIOP 1.2 byte order markers (since CORBA 2.4-5)
jacorb.use_bom=off

# add additional IIOP 1.0 profiles even if we are using IIOP 1.2
jacorb.giop.add_1_0_profiles=off

###########################################
#                                         #
#         Socket Factories                #
#                                         #
###########################################

# A factory design pattern is used for the creation of sockets and server
# sockets.
# The jacorb.net.socket_factory property can be used to configure
# a socket factory that must implement the operations defined in the
# interface org.jacorb.orb.factory.SocketFactory.
# The jacorb.net.server_socket_factory property can be used to configure a
# server socket factory that must implement the operations defined in the
# interface org.jacorb.orb.factory.ServerSocketFactory.
#
#jacorb.net.socket_factory=org.jacorb.orb.factory.DefaultSocketFactory
#jacorb.net.server_socket_factory=org.jacorb.orb.factory.DefaultServerSocketFac
tory
#
# An additional socket factory is supported that allows for the configuration
# of maximum and minimum port numbers that can be used. This can be used to
# enable firewall traversal via a fixed port range. To use this socket factory
# configure the following two properties.
#
#jacorb.net.socket_factory.port.min
#jacorb.net.socket_factory.port.max

###########################################
#                                         #
#         BiDirectional GIOP              #
#                                         #
###########################################

# uncomment this initializer if you want to use BiDirectional GIOP

#org.omg.PortableInterceptor.ORBInitializerClass.bidir_init=org.jacorb.orb.conn
```

```
ection.BiDirConnectionInitializer


###########################################
#                                         #
#       Proxy address in IOR              #
#                                         #
###########################################

#
# with these two properties it is possible to
# tell the ORB what IP/port IORs should contain,
# if the ServerSockets IP/port can’t be used
# (e.g. for traffic through a firewall).
#
# WARNING: this is just "dumb" replacing, so you
# have to take care of your configuration!
#

#jacorb.ior_proxy_host=1.2.3.4
#jacorb.ior_proxy_port=4711


###########################################
#                                         #
#   The Object Adapter Internet Address   #
#                                         #
###########################################

# IP address on multi-homed host (this gets encoded in
# object references). NOTE: Adresses like 127.0.0.X
# will only be accessible from the same machine!
#OAIAddr=1.2.3.4
#OAPort=4711

################################
#                              #
#   Appligator Configuration   #
#                              #
################################
# if your applets don’t need appligator, switch this off
jacorb.use_appligator=off



############################
#                          #
#   Default Interceptors   #
#   Please leave them in!  #
#                          #
############################
org.omg.PortableInterceptor.ORBInitializerClass.standard_init=org.jacorb.orb.st
andardInterceptors.IORInterceptorInitializer



###############################################
#                                             #
#   Implementation Repository Configuration   #
#                                             #
###############################################
# Switch off to avoid contacting the ImR on every server start-up
jacorb.use_imr=off

# if set to "on", servers that don't already have an entry on their
# first call to the imr, will get automatically registered. Otherwise,
# an UnknownServer exception is thrown.
jacorb.imr.allow_auto_register=off

# if set to "on", the imr will try to "ping" every object reference,
# that it is going to return. If the reference is not alive, TRANSIENT
# is thrown.
jacorb.imr.check_object_liveness=off

ORBInitRef.ImplementationRepository=http://www.x.y.z/~user/ImR_Ref

jacorb.imr.table_file=Z:\table.dat
jacorb.imr.backup_file=z:\backup.dat
jacorb.imr.ior_file=/home/bwana/brose/public_html/ImR_Ref
jacorb.imr.timeout=
jacorb.imr.no_of_poas=
jacorb.imr.no_of_servers=

# how many millis should the imr wait, until a connection from an
# application client is terminated. Default is 2000.
jacorb.imr.connection_timeout=2000

# the implementation name, should be set to a different
# name in the code of persistent servers
jacorb.implname=StandardImplName

#
# This is supposed to be a generic startup string for everything
# that calls Runtime.exec(). Might be replaced by jaco[.bat].
#
jacorb.java_exec=java -Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton
```

```
#########################
#                       #
#   SSL Configuration   #
#                       #
#########################

#
# The port number used by SSL, will be dynmically assigned
# by default
#

#OASSLPort=4711

# This interceptor must be set if programs need access to
# certificates using the CORBA Security API, SSL works also
# without this interceptor

#org.omg.PortableInterceptor.ORBInitializerClass.ForwardInit=org.jacorb.securit
y.ssl.SecurityServiceInitializer


# qualified classname of access decision object
jacorb.security.access_decision=org.jacorb.security.level2.AccessDecisionImpl

# list of qualified classnames of principal authenticator objects,
# separated by commas (no whitespaces!). The first entry (that can
# be successfully created) will be available through the
# principal_authenticator property.
jacorb.security.principal_authenticator=org.jacorb.security.level2.PrincipalAut
henticatorImpl

# the qualified classname of the ssl socket factory class
#jacorb.ssl.socket_factory=org.jacorb.security.ssl.sun_jsse.SSLSocketFactory
jacorb.ssl.socket_factory=org.jacorb.security.ssl.iaik.SSLSocketFactory

# the qualified classname of the ssl server socket factory class
#jacorb.ssl.server_socket_factory=org.jacorb.security.ssl.sun_jsse.SSLServerSoc
ketFactory
jacorb.ssl.server_socket_factory=org.jacorb.security.ssl.iaik.SSLServerSocketFa
ctory

# exchange ssl client server roles to enforce client authentication, but
# attention: this causes problems with peers that not prepared to handle
# this role change
jacorb.security.change_ssl_roles=off

# IIOP/SSL parameters (numbers are hex values, without the leading "0x"):
```

```
# NoProtection = 1
# EstablishTrustInClient = 40
# EstablishTrustInTarget = 20
# mutual authentication = 60
# please see the programming guide for more explanation

jacorb.security.support_ssl=off

jacorb.security.ssl.client.supported_options=0
jacorb.security.ssl.client.required_options=0

jacorb.security.ssl.server.supported_options=0
jacorb.security.ssl.server.required_options=0

#
# If set, the following two values will be placed in the IOR, if
# "corbaloc:ssliop" ssliop.
#
# If not set, only EstablishTrustInTarget is used for both supported
# and required options. EstablishTrustInClient is not set, and the
# rest of the Association Options aren't currently used anyway.
#jacorb.security.ssl.corbaloc_ssliop.supported_options=0
#jacorb.security.ssl.corbaloc_ssliop.required_options=0

# The name and location of the keystore. This may be absolute or
# relative to the home directory.
#
# NOTE (for Sun JSSE users): The "javax.net.ssl.trustStore[Password]"
# properties don't seem to take effect, so you may want to add trusted
# certificates to "normal" keystores. In this case, please set the
# property "jacorb.security.jsse.trustees_from_ks"is to "on", so trusted
# certificates are taken from the keystore instead of a dedicated
# truststore.
jacorb.security.keystore=
jacorb.security.keystore_password=


#
# IAIK specific settings
#

# files with public key certs of trusted CAs
#
# WARNING: If no CA certs are present, the IAIK chain verifier will
# accept ALL otherwise valid chains!
#
jacorb.security.trustees=

# the name of the default key alias to look up in the keystore
```

```
jacorb.security.default_user=
jacorb.security.default_password=

# have iaiks ssl classes print debug output to stdout
jacorb.security.iaik_debug=off

#
# Sun JSSE specific settings
#
# Use the keystore to take trusted certs from.
jacorb.security.jsse.trustees_from_ks=off

# A comma-separated (no whitespaces!) list of cipher suite names. See
# the JSSE docs on how to obtain the correct cipher suite strings
jacorb.security.ssl.server.cipher_suites=
jacorb.security.ssl.client.cipher_suites=

########################
#                      #
#   POA Configuration  #
#                      #
########################

# displays a GUI monitoring tool for servers
jacorb.poa.monitoring=off

# thread pool configuration for request processing
jacorb.poa.thread_pool_max=20
jacorb.poa.thread_pool_min=5

# if set, request processing threads in thePOA
# will run at this priority. If not set or invalid,
# MAX_PRIORITY will be used.
#jacorb.poa.thread_priority=

# size of the request queue, clients will receive Corba.TRANSIENT
# exceptions if load exceeds this limit
jacorb.poa.queue_max=100



########################################
#                                      #
#   Trader configuration, please see   #
#   src/trading/README.PROPERTIES for  #
#   explanation                        #
#                                      #
########################################
```

```
jtrader.util.max_threads=10
jtrader.util.min_threads=1
jtrader.util.query_timeout=5000
jtrader.impl.cache_max=100

# boolean values, e.g. true / false
#jtrader.modifiable_properties=
#jtrader.dynamic_properties=
#jtrader.proxy_offers=

jtrader.debug=false
jtrader.debug_verbosity=3

#integer values
jtrader.def_search_card=
jtrader.max_search_card=
jtrader.def_match_card=
jtrader.max_match_card=
jtrader.def_return_card=
jtrader.max_return_card=
jtrader.max_list=
jtrader.def_hop_count=
jtrader.max_hop_count=

#FollowOptions
#always=2
#if_no_local=1
#local_only=0
jtrader.def_follow_policy=
jtrader.max_follow_policy=
jtrader.max_link_follow_policy=

# any other custom properties can be added here.
# These are available through the API (call
# jacorb.orb.Environment.getProperty())
```

Build XML

This file enables you to build the sample JacORB application referred to in "Building and Running the JacORB Request Broker" in "Using JacORB With the BEA WebLogic Adapter for CORBA" in Appendix A, "Using CORBA Implementations With the Adapter."

Listing B-40 Build XML

```
<?xml version="1.0"?>

<project name="bea" default="all" basedir="../..">

  <!-- ==================================================== -->
  <!--            build file                                -->
  <!-- ==================================================== -->

  <target name="init">
   <property name="name" value="club"/>
   <property name="dirs.base" value="${basedir}"/>
   <property name="classdir" value="${dirs.base}/classes"/>
   <property name="lib" value="${dirs.base}/lib"/>
   <property name="include" value="${dirs.base}/idl"/>
   <property name="idlflags" value="-I${include}/omg -ir -d
${dirs.base}/bea/${name}/generated"/>
  </target>

  <target name="all" depends="init,idl">
   <javac srcdir="${dirs.base}/bea/${name}/generated"
            destdir="${classdir}"
          includes="**/*.java"
           />

   <javac srcdir="${dirs.base}/bea/${name}"
            destdir="${classdir}"
          includes="*.java"
           />
  </target>

  <target name="idl" depends="init">
      <java classname="org.jacorb.idl.parser"
             fork="yes"
             classpath="${lib}/idl.jar;${java.class.path}">
          <arg line="${idlflags}
            ${dirs.base}/bea/${name}/clubmed.idl"/>
       </java>
   </target>


   <target name="clean" depends="init">
           <delete dir="${classdir}/bea/${name}"/>
           <delete dir="${dirs.base}/bea/${name}/generated"/>
   </target>

</project>
```

ClubMed.IDL File

The `clubmed.idl` file displays the Interface Repository in IDL.

Listing B-41 ClubMed.IDL File

```
// ClubMed.idl
module bea
{
module club
{
  struct pricesStruct
  {
    string date;
    short  adultFare;
    short  childFare;
  };
  typedef sequence<pricesStruct> pricesSeq;

  typedef sequence<string> clubNamesSeq;

  typedef long resvNo;

  typedef pricesStruct pricesArray[4];

  struct resvStruct
  {
    string fname;
    string lname;
    string weekDate;
    string address;
    string city;
    string state;
    string zip;
    string phone;
    string totalFare;
    short  partyAdults;
    short  partyChildren;
    string date;
  };

  interface ClubMed
  {
    exception ClubException
    {
      string reason;
    };
    pricesSeq    getClubPrices(in string club);
    pricesArray  getClubPricesAsArray(in string club);
    resvNo       addReservation(in string club, in resvStruct resvData);
    resvStruct   getReservation(in resvNo resv);
    clubNamesSeq getClubNames();
 };

};
};
```

ClubServer.Java File

The `ClubServer.java` file displays the Interface Repository in Java.

Listing B-42 ClubServer.Java File

```
// The package containing our stubs.
package bea.club;

import java.util.*;

// HelloServer will use the naming service.
import org.omg.CosNaming.*;
// All CORBA applications need these classes.
```

```
import org.omg.CORBA.*;

public class ClubServer
{
  public static void main(String args[]) {
    // Create and initialize the ORB
    Properties props = new Properties();
    props.setProperty("org.omg.CORBA.ORBClass", "org.jacorb.orb.ORB");
    props.setProperty("org.omg.CORBA.ORBSingletonClass",
"org.jacorb.orb.ORBSingleton");
    ORB orb = ORB.init(args, props);
  try {
      org.omg.PortableServer.POA poa =

org.omg.PortableServer.POAHelper.narrow(orb.resolve_initial_references("RootPOA
"));

    poa.the_POAManager().activate();

    org.omg.CORBA.Object o = poa.servant_to_reference(new ClubsServant());

    // use the naming service

  NamingContextExt nc =

NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService"));
  nc.bind( nc.to_name("bea.clubmed"), o);

      orb.run();

    } catch(Exception e) {
      System.err.println("ERROR: " + e);
      e.printStackTrace(System.out);
    }
  }
}

class ClubsServant extends ClubMedPOA
{
  private int resNum = 999;
  private Map map = new HashMap();

  public pricesStruct[] getClubPrices (String club) {
    pricesStruct[] prices = new pricesStruct[4];
    prices[0] = new pricesStruct("2002-11-23", (short)88, (short)50);
    prices[1] = new pricesStruct("2002-11-24", (short)88, (short)50);
    prices[2] = new pricesStruct("2002-11-25", (short)80, (short)45);
    prices[3] = new pricesStruct("2002-11-26", (short)80, (short)45);
    return prices;
  }

  public pricesStruct[]  getClubPricesAsArray(String club) {
    pricesStruct[] prices = new pricesStruct[4];
    prices[0] = new pricesStruct("2002-11-23", (short)88, (short)50);
    prices[1] = new pricesStruct("2002-11-24", (short)88, (short)50);
    prices[2] = new pricesStruct("2002-11-25", (short)80, (short)45);
    prices[3] = new pricesStruct("2002-11-26", (short)80, (short)45);
    return prices;
  }

  public String[] getClubNames() {
    return new String[] {"BAMBU", "NAEBO", "RIO PALACE" };
  }

  public int addReservation (String club, resvStruct resvData) {
    System.out.println(club);
    System.out.println(resvData.address);
    System.out.println(resvData.city);
    System.out.println(resvData.date);
    System.out.println(resvData.fname);
    System.out.println(resvData.lname);
    System.out.println(resvData.partyAdults);
    System.out.println(resvData.partyChildren);
    System.out.println(resvData.phone);
    System.out.println(resvData.weekDate);
    System.out.println(resvData.zip);
    System.out.println(resvData.totalFare);
    map.put(new Integer(++resNum), resvData);
    return resNum;
  }

  public resvStruct getReservation (int resv) {
    resvStruct resvData = (resvStruct)map.get(new Integer(resv));
    System.out.println("Returning: " + resvData);
    return resvData;
  }

  public boolean cancelReservation (int resv) {
    return false;
  }


}
```