SYBASE®

Cookbook

EAServer

5.0

DOCUMENT ID: DC38038-01-0500-01

LAST REVISED: December 2003

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book ..... **v**

CHAPTER 1 **Getting Started with EAServer** **1**
- Understanding EAServer 1
- Starting the server 3
  - Starting servers on Windows platforms 3
  - Starting the server on UNIX platforms 4
- Using EAServer Manager 4
  - Starting Sybase Central on Windows 5
  - Starting Sybase Central on UNIX platforms 5
  - Connecting to the EAServer Manager plug-in 5
- Shutting down, restarting, or refreshing the server 6

CHAPTER 2 **Creating CORBA Java Components and Clients** **9**
- Overview of the sample application 9
- Tutorial requirements 10
- Java compiler scripts 10
- Creating the application 10
  - Start EAServer and EAServer Manager 11
  - Define a package, component, and method 11
  - Generate stubs and skeletons 13
  - Write the server-side code 14
  - Create the client program 15
  - Run the client program 19
- Creating an applet client 20
  - Verify that your browser can run JDK 1.2 applets 20
  - Prepare your EAServer installation to run applets 22
  - Write the applet code 22
  - Create an HTML page to run the applet 26
  - Run the applet 28

CHAPTER 3 **Creating C++ Components and Clients** **29**
- Overview of the sample application 29

Tutorial requirements 30
Creating the application 30
Verify your environment 30
Start EAServer and EAServer Manager 32
Define a package, component, and method 32
Generate stubs, skeletons, and implementation templates 35
Write the server-side code 35
Write the client-side code 37
Compile the client executable 41
Run the client executable 43

CHAPTER 4 Creating Enterprise JavaBeans Components and Clients 45
Overview of the sample components 45
Tutorial requirements 45
Creating the application 46
Start EAServer and EAServer Manager 46
Create the package and components 47
Create the glossary database and connection cache 51
Create the client application 54
Run the client application 54

CHAPTER 5 Using the EAServer Samples 57
Samples in the EAServer installation 57
Java samples 57
C++ samples 60
Java Pet Store 61
PowerBuilder samples 62
Web Services Toolkit samples 62
Samples on the Sybase Web site 62

Index 63

About This Book

Subject

This book contains tutorials and explains how to use the sample applications included with your EAServer software.

Audience

This book is intended for new EAServer programmers. You should be familiar with your development language of choice.

How to use this book

This book includes these chapters:

- Chapter 1, “Getting Started with EAServer,” describes how to start the preconfigured server and connect to it with EAServer Manager before running the tutorials.
- Chapter 2, “Creating CORBA Java Components and Clients,” describes how to create, implement, and run a Java component and client using the EAServer CORBA component model.
- Chapter 3, “Creating C++ Components and Clients,” describes how to create, implement, and run a C++ component and client using the EAServer CORBA component model.
- Chapter 4, “Creating Enterprise JavaBeans Components and Clients,” describes how to create an EJB session bean and entity bean, and call them from an EJB client.
- Chapter 5, “Using the EAServer Samples,” describes the samples included in the EAServer installation and where to find additional samples on the World Wide Web.

Related documents

Core EAServer documentation The core EAServer documents are available in HTML format in your EAServer software installation, and in PDF and DynaText format on the *Technical Library* CD.

What's New in EAServer summarizes new functionality in this version.

The *EAServer Feature Guide* explains application server concepts and architecture, such as supported component models, network protocols, server-managed transactions, and Web applications.

The *EAServer System Administration Guide* explains how to:

- Start the preconfigured Jaguar server and manage it with the EAServer Manager plug-in for Sybase Central™

- Create, configure, and start new application servers
- Define connection caches
- Create clusters of application servers to host load-balanced and highly available components and Web applications
- Monitor servers and application components
- Automate administration and monitoring tasks with jagtool

The *EAServer Programmer's Guide* explains how to:

- Create, deploy, and configure components and component-based applications
- Create, deploy, and configure Web applications, Java servlets, and JavaServer Pages
- Use the industry-standard CORBA and Java APIs supported by EAServer

The *EAServer Web Services Toolkit User's Guide* describes Web services support in EAServer, including:

- Support for standard Web services protocols such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Uniform Description, Discovery, and Integration (UDDI)
- Administration tools for deployment and creation of new Web services, WSDL document creation, UDDI registration, and SOAP management

The *EAServer Security Administration and Programming Guide* explains how to:

- Understand the EAServer security architecture
- Configure role-based security for components and Web applications
- Configure SSL certificate-based security for client connections using the Security Manager plug-in for Sybase Central
- Implement custom security services for authentication, authorization, and role membership evaluation
- Implement secure HTTP and IIOP client applications
- Deploy client applications that connect through Internet proxies and firewalls

The *EAServer Performance and Tuning Guide* describes how to tune your server and application settings for best performance.

The *EAServer API Reference Manual* contains reference pages for proprietary EAServer Java classes, ActiveX interfaces, and C routines.

The *EAServer Troubleshooting Guide* describes procedures for troubleshooting problems that EAServer users may encounter. This document is available only online; see the EAServer Troubleshooting Guide at http://www.sybase.com/detail?id=1024509.

Message Bridge for Java™ Message Bridge for Java simplifies the parsing and formatting of structured documents in Java applications. Message Bridge allows you to define structures in XML or other formats, and generates Java classes to parse and build documents and messages that follow the format. The *Message Bridge for Java User's Guide* describes how to use the Message Bridge tools and runtime APIs. This document is included in PDF and DynaText format on your *EAServer 5.0 Technical Library* CD.

Adaptive Server Anywhere documents EAServer includes a limited-license version of Adaptive Server Anywhere for use in running the samples and tutorials included with EAServer. Adaptive Server Anywhere documents are available on the Sybase Web site at http://sybooks.sybase.com/aw.html.

jConnect for JDBC documents EAServer includes the jConnect™ for JDBC™ driver to allow JDBC access to Sybase database servers and gateways. The *Programmer's Reference jConnect for JDBC* is available on the Sybase Web site at http://sybooks.sybase.com/jc.html.

Conventions

The formatting conventions used in this manual are:

Formatting example	To indicate
commands and methods	When used in descriptive text, this font indicates keywords such as: • Command names used in descriptive text • C++ and Java method or class names used in descriptive text • Java package names used in descriptive text • Property names in the raw format, as when using jagtool to configure applications rather than EAServer Manager
variable, *package*, or *component*	Italic font indicates: • Program variables, such as *myCounter* • Parts of input text that must be substituted, for example: `Server.log` • File names • Names of components, EAServer packages, and other entities that are registered in the EAServer naming service

Formatting example	To indicate
File | Save	Menu names and menu items are displayed in plain text. The vertical bar shows you how to navigate menu selections. For example, File | Save indicates "select Save from the File menu."
`package 1`	Monospace font indicates: • Information that you enter in EAServer Manager, a command line, or as program text • Example program fragments • Example output fragments

Accessibility features

EAServer 5.0 has been tested for compliance with U.S. government Section 508 Accessibility requirements. The online help for this product is also provided in HTML, JavaHelp, and Eclipse help formats, which you can navigate using a screen reader.

EAServer Manager supports working without a mouse. For more information, see "Keyboard navigation" in Chapter 2, "Sybase Central Overview," in the *EAServer System Administration Guide*.

The WST plug-in for Eclipse supports accessibility features for those that cannot use a mouse, are visually impaired or have other special needs. For information about these features refer to Eclipse help:

1 Start Eclipse

2 Select Help | Help Contents

3 Enter Accessibility in the Search dialog box

4 Select Accessible user interfaces or Accessibility features for Eclipse

---

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For additional information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

Other sources of information

Use the Sybase Getting Started CD, the Sybase Technical Library CD and the Technical Library Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the Technical Library CD. It is included with your software. To read or print documents on the Getting Started CD you need Adobe Acrobat Reader (downloadable at no charge from the Adobe Web site, using a link provided on the CD).
- Technical Library CD contains product manuals and is included with your software. The DynaText browser (downloadable from Product Manuals at http://www.sybase.com/detail/1,3693,1010661,00.html) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.
- Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, you will find links to the Technical Documents Web site (formerly known as Tech Info Library), the Solved Cases page, and Sybase/Powersoft newsgroups.

  To access the Technical Library Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Getting Started with EAServer

Topic	Page
Understanding EAServer	1
Starting the server	3
Using EAServer Manager	4
Shutting down, restarting, or refreshing the server	6

Understanding EAServer

This section explains some of the basic concepts and terminology associated with developing component-based Jaguar applications in a three-tier environment. It is intended primarily to provide you with enough information to complete the tutorials and begin using EAServer Manager. For detailed information on Jaguar application development, see the *EAServer Programmer's Guide*. For details on the EAServer architecture, see the *EAServer Feature Guide*.

EAServer implements a *three-tier* or *multitier* distributed computing architecture. In this model, three distinct elements work together to give users access to data:

- A user interface, which can be a standalone client program or a Web application run in the user's Web browser. Web applications can use JSPs, Java servlets, or Java applets to manage presentation and interaction with the end user. Clients use stubs to communicate with application components running in the middle tier.
- Middle-tier components, which access data from one or more databases, apply business logic, and return results to the client for display.
- The back-end database, which hosts application data.

An EAServer application consists of one or more packages and a client user interface to interact with end users. Packages consist of components, and components are made up of one or more methods.

- In EAServer, a *component* is simply an *application object* that consists of one or more methods. EAServer can host, manage, and execute components such as Enterprise JavaBeans, CORBA-compliant Java and C++ components, and ActiveX nonvisual components. Components typically execute business logic, access data sources, and return results to the client.
- A *package* is a collection of components that work together to provide a service or some aspect of your application's business logic. A package defines a *boundary of trust* within which components can easily communicate. Each package acts as a unit of distribution, simplifying deployment and management of related components.
- A *stub* is a Java or C++ class generated by EAServer Manager. The stub acts as a *proxy object* for a Jaguar component. A stub is compiled and linked with your Java applets or client application. A stub communicates with Jaguar to instantiate and invoke a method on a component in the middle tier. Stubs make a remote Jaguar component appear local to the client.
- A *skeleton* is a Java or C++ class generated by EAServer Manager. The skeleton acts as the interface between the EAServer runtime and the user code that implements the component.
- A Web application allows you to deploy Java servlets, JavaServer Pages (JSPs), and related Web files as a unit. Servlets and JSPs in a Web application can invoke EAServer components, allowing you to develop user interfaces that run over the Web without using Java applets.
- A *server* is an operating system process that provides the runtime environment to execute components and Web applications in response to client requests. The tutorials use the preconfigured server named Jaguar, though you can define and configure your own servers in EAServer Manager.
- EAServer supports several network communication protocols. These protocols are of interest for the tutorials:
  - *IIOP* is a standard CORBA protocol for component invocations. Standalone client applications and client applets use IIOP to invoke components. You do not need to know IIOP; the generated stubs take care of all network communication.

- *HTTP* is the standard Web protocol for file downloads and form requests. Web browser clients use HTTP to invoke servlets, JSPs, and to download Web pages.

Each supported protocol requires a listener to be associated with the server in EAServer Manager. A listener defines the port number, host name, and protocol for client connections to the server. You do not need to configure any listeners to run the tutorials, unless you plan to run client applications on a different machine than the server. In that case, you must change the listener host names from the default, localhost, to the server machine name.

For information on listeners and other supported protocols, see "Configuring listeners" in the *EAServer System Administration Guide*. For a tutorial that uses the secure versions of IIOP and HTTP, see the *EAServer Security Administration and Programming Guide*.

Starting the server

Before you can work through the tutorials, run the sample applications, or develop EAServer applications, you must start the preconfigured Jaguar server.

Note These procedures start the preconfigured server with the default configuration, which is adequate for running the tutorials. There are many other options for server start-up not discussed here, such as selecting a different Java virtual machine or running EAServer as a Windows service. For these details and more, see "Starting the server" in the *EAServer System Administration Guide*.

Starting servers on Windows platforms

You can start the server from the command line or using the Start button.

❖ **Using the Start button**

1 Select Start | Programs | Sybase | EAServer 5.0 | Jaguar Server.

2 The server starts and runs in a console window.

❖ **Using the command line**

1 Change to the EAServer *bin* subdirectory, and run the serverstart command. For example:

```
c:
cd \Program Files\Sybase\EAServer 5.0\bin
start serverstart
```

2 The server starts and runs in a new console window. To run the server in the same window as your command prompt, omit start from the command.

Starting the server on UNIX platforms

Start the server from the command line.

❖ **Starting the server**

- Change to the EAServer *bin* directory, and run serverstart.sh. For example:

```
cd /opt/Sybase/EAServer 5.0/bin
serverstart.sh
```

The server starts and runs as a foreground process in the current console window. To start the server in a separate window, specify the -xterm option. For example:

```
cd /opt/Sybase/EAServer 5.0/bin
serverstart.sh -xterm
```

The server starts in a new xterm terminal window. X Windows must be installed and configured to use this option. In other words, the xterm command must successfully launch a terminal window in the shell where you start the server.

Using EAServer Manager

EAServer Manager runs as a Sybase Central Java plug-in. EAServer Manager allows you to connect to EAServer and configure servers, packages, components, and Web applications. For more information on EAServer Manager and Sybase Central, see Chapter 2, “Sybase Central Overview,” in the *EAServer System Administration Guide*.

To use EAServer Manager, you must start Sybase Central, then connect to the EAServer Manager plug-in, which in turn connects you to a running EAServer instance. The server must be running before you can connect to it.

Starting Sybase Central on Windows

You can start EAServer Manager from the command line or using the Start button.

❖ **Using the Start button**

1 Select Start | Programs | Sybase | EAServer 5.0 | EAServer Manager.

2 Connect to the EAServer Manager plug-in as described below.

❖ **Using the command line**

1 Change to the EAServer *bin* subdirectory, and run the jagmgr command. For example:

```
c:
cd \Program Files\Sybase\EAServer 5.0\bin
start jagmgr
```

2 Connect to the EAServer Manager plug-in as described below.

Starting Sybase Central on UNIX platforms

Start Sybase Central on the command line.

❖ **Starting Sybase Central**

1 Change to the EAServer *bin* directory, and run jagmgr. For example:

```
cd /opt/Sybase/EAServer 5.0/bin
jagmgr &
```

2 Connect to the EAServer Manager plug-in as described below.

Connecting to the EAServer Manager plug-in

After starting Sybase Central, connect to the EAServer Manager plug-in, which establishes your management connection to EAServer.

Administrative privileges required
You must have EAServer administrative privileges to run the tutorials. For a new installation, use the jagadmin user name, and leave the password blank. If you are using a server installed and administered by someone else, you must have access to the jagadmin password, or to another account in the role named Admin Role.

❖ **Connecting to EAServer Manager**

1 Select Tools | Connect | EAServer Manager.

2 On the Login screen:

   a Enter `jagadmin` as the user name, or specify another user name that is in the role Admin Role.

   b Enter the password for the account you are using.

   c Specify the host name or IP address of the Jaguar server. If you have not changed the Jaguar server's listener properties, enter `localhost`.

   d Specify the Jaguar server's IIOP port number. If you have not changed the Jaguar server's listener properties, enter `9000`.

   e Click Connect.

3 The EAServer Manager icon expands to include folders for Agents, Clusters, and so forth.

Shutting down, restarting, or refreshing the server

Some tutorial steps require that you restart, refresh, or shut down the server. From EAServer Manager, you can:

- Shut down a server, which means you must restart the server manually as described in "Starting the server" on page 3, then reconnect to EAServer Manager as described in "Connecting to the EAServer Manager plug-in" on page 5.
- Refresh a server, which reinitializes all installed components and Web applications. Newly installed components and Web applications become available to clients, and running components and Web applications are reinitialized.

- Restart a server, which shuts down the server process and automatically restarts a new process. The EAServer Manager plug-in automatically connects to the new server process, unless you have changed the host name or port number properties for the IIOP listener.

You must be connected to a server to refresh, shut down, or restart that server. You can also shut down servers outside of EAServer Manager as described in "Shutting down without using EAServer Manager" on page 7.

❖ **Shutting down, refreshing, or restarting using EAServer Manager**

You must be connected to the target server to perform these procedures.

1 Expand the Servers folder, then highlight the icon for the server to which you are connected, for example, Jaguar.

2 Use the File menu to perform the required operation:

- To restart, select File | Shutdown and Start.
- To shut down, select File | Shutdown.
- To refresh, select File | Refresh.

❖ **Shutting down without using EAServer Manager**

You can use this procedure if you are running the server in the foreground in its own console or terminal window. If you are running the server in the background or as a Windows service, shut down using the procedures described in the *EAServer System Administration Guide*.

- Use the system-specific window control to close the window in which the server is running. For example:
  - Right-click on the title bar and choose Close, or
  - Click the close icon in the title bar.

  On UNIX you can also kill the server process by placing the input cursor inside the window's text area and pressing Ctrl+C.

CHAPTER 2

Creating CORBA Java Components and Clients

In this tutorial, you will create a CORBA Java component, install it in EAServer, and create a CORBA Java client that connects to EAServer and calls a method in the component.

Topic	Page
Overview of the sample application	9
Tutorial requirements	10
Java compiler scripts	10
Creating the application	10
Creating an applet client	20

For more information

For complete information on creating Java components and Java clients, see these chapters in the *EAServer Programmer's Guide*:

- Chapter 11, "Creating CORBA Java Components"
- Chapter 12, "Creating CORBA Java Clients"

Overview of the sample application

The application performs the following steps:

1 The client-side application, developed with Java, instantiates the middle-tier Java component, JavaArithmetic.

2 The client calls the multiply method in JavaArithmetic.

3 The multiply method computes the product of the input values, then returns the result.

4 The client application displays the result for the end user.

Tutorial requirements

To create the tutorial application, you need:

- The EAServer software

  The *EAServer Installation Guide* for your platform describes how to install the software.

- Java development environment

  The tutorial steps use the JDK software that is included with your EAServer installation. You can also use JBuilder or any development tool that is compatible with JDK 1.3.

Java compiler scripts

This tutorial assumes that you are using the Java compiler that is provided by your EAServer installation; however, you can use any compiler as long as it produces bytecode that is compatible with JDK 1.3 or later.

EAServer provides a compilation script in the *bin* subdirectory to compile Java files using the JDK included with EAServer:

Platform	Compilation script
Windows	*bin\jc.bat*
UNIX (all)	*bin/jc*

The compilation steps in the tutorial use these scripts. You can also use any IDE that is compatible with JDK 1.3 or later.

Creating the application

To create and run the sample application:

1 Start EAServer and EAServer Manager.

2 Define a package, component, and method.

3 Generate stubs and skeletons.

4 Write the server-side code.

5 Create the client program.

6 Run the client program.

Start EAServer and EAServer Manager

❖ **Starting EAServer**

- If EAServer is not already running, follow the instructions under "Starting the server" on page 3 to start the server.

❖ **Starting EAServer Manager**

- If EAServer Manager is not already running, start it as described in "Using EAServer Manager" on page 4.

Define a package, component, and method

This section shows you how to use EAServer Manager to create the package, component, and method for the sample application.

For complete information on creating packages, components, and methods, see Chapter 5, "Defining Component Interfaces," in the *EAServer Programmer's Guide*.

Define a new package

In EAServer, a package is a unit of deployment for a group of components that perform related tasks. Before a component can be instantiated by clients, it must be installed in a package, and that package must be installed in the server. The steps below create the package and component within the predefined "Jaguar" server to satisfy these requirements.

❖ **Creating the *Tutorial* package if it does not exist**

1 In EAServer Manager, expand the servers folder, then expand the Jaguar server icon.

2 Expand the Installed Packages folder. If the Tutorial package is displayed, skip to "Define and install a new component" on page 12.

3 Highlight the Installed Packages folder, and select File | Install Package.

In the Package wizard, select Create and Install a New Package.

For the package name, enter `Tutorial`.

4 Click Create New Package.

You see the Package Properties window.

5 Click OK.

Define and install a new component

❖ **Defining the new component**

1 Click the Tutorial package.

2 Select File | New Component.

3 In the Define New Component wizard, select Define New Component and click Next.

4 For the component name, enter `JavaArithmetic`.

5 Click Finish. You see the Component Properties window.

6 Select the General tab. Fill in the fields as follows:

Field	Value
Description	Tutorial Java component
Component Type	Java - CORBA
Java Class	Sample.Intro.JavaArithmetic.JavaArithmeticImpl

7 Leave the remaining fields at their default settings.

8 Click OK.

Define the multiply method

The component interface will have one method, multiply.

❖ **Defining the component interface**

1 Expand the Tutorial package. Double-click the JavaArithmetic component to show the Roles and Interfaces folders beneath it.

2 Double-click the Interfaces folder, and highlight the Tutorial::JavaArithmetic interface. If you do not see this interface, install it as follows:

a Highlight the Interfaces folder and select File | Add Interfaces ...

b In the Install Interface dialog box, highlight Tutorial::JavaArithmetic in the Selected to Install table, then click Install.

c Highlight the Tutorial::JavaArithmetic interface that is now displayed under the Interfaces folder.

3 Select File | New Method.

4 Assign the name multiply to the method.

5 Click Create New Method.

You see the Method Properties window.

6 In the Return field, select double as the method's return type.

7 Beneath the empty parameter list, click Add to add a parameter. In the New Parameter dialog:

- For the parameter name, enter `m1`.
- For Mode, select `in`.
- For Type, select `double`.

8 Click OK to close the New Parameter dialog box.

9 Repeat steps 7 and 8 to add a second parameter named `m2` with a Type of double.

10 Click OK to close the Method Properties dialog box.

Generate stubs and skeletons

Once you have created the package, component, and methods, you generate the stub and skeleton files for the component. The client-side application uses the stubs to invoke the server-side component methods. The skeleton acts as an interface between EAServer and your component methods.

❖ **Generating the stub and skeleton files for the component**

1 Click the Tutorial package and select the JavaArithmetic component.

2 Select File | Generate Stub/Skeleton.

3 Select Generate Stubs, then select Generate Java Stubs. Fill in the Java Stubs fields as follows:

a Select CORBA from the drop-down list.

b Select Generate Java Files.

c Select Compile Java Stubs.

d Leave the Java Code Base field at the default:

`%JAGUAR%\html\classes`	For Windows
`$JAGUAR/html/classes`	For UNIX

e Deselect Generate C++ Stubs.

4 Click Next to display the skeleton generation options and configure the settings as follows:

a Select Generate Skeletons on Client.

b Leave the Java Code Base field at the default value:

`%JAGUAR%\java\classes`	For Windows
`$JAGUAR/java/classes`	For UNIX

c Deselect "Compile Java Skeletons" (you cannot compile now because the component implementation source file is not ready).

5 Click Finish.

Write the server-side code

At this point, EAServer Manager has created server-side implementation files in the following directory under your EAServer installation:

```
java/classes/Sample/Intro/JavaArithmetic
```

The implementation template file is *JavaArithemeticImpl.Java.new* and the skeleton is called *_sk_Tutorial_JavaArithmetic.java*.

❖ Completing the component implementation

1 Rename *JavaArithemeticImpl.Java.new* to *JavaArithmeticImpl.Java* (that is, delete the *.new* extension). Open the renamed file in a text editor, then find the definition of the multiply method. Change the definition so that it matches the one below:

```
double multiply
    (double m1,
    double m2)
 {
      return m1 * m2;
```

```
}
```

2 Save your changes.

3 Compile the component skeleton and implementation files using a JDK 1.3 or later compiler—for example, if you are using Windows:

```
cd %JAGUAR%\java\classes\Sample\Intro\JavaArithmetic
%JAGUAR%\bin\jc.bat *.java
```

Or, if you are using UNIX:

```
cd $JAGUAR/java/classes/Sample/Intro/JavaArithmetic
$JAGUAR/bin/jc *.java
```

Create the client program

In the *html\classes* subdirectory of your EAServer installation, create a new directory called *TutorialApps* if it does not exist. In this directory, create the Java file below as *JAConsole.java*.

This is a simple command-line application that:

- Connects to EAServer.
- Creates an authenticated session. Edit the user name and password in the source code if your server is configured to validate passwords and role memberships. By default, authentication and role checking are disabled in the server.
- Creates a proxy for the component.
- Calls the component multiply method.

You can find a copy of *JAConsole.java* in the *html/docs/tutorial/java-corba* subdirectory of your EAServer installation. Here is the source for *JAConsole.java*:

```
//
// This is a sample command-line Java application that
// invokes the JavaArithmetic component created in the EAServer
// Java component tutorial.
//
// Usage:
//    arith iiop://<host>:<port>
//
//    Where:
//
```

```
//       <host> is the host name or IP address of the server machine.
//
//       <iiop-port> is the server's IIOP port (9000 in the
//         default configuration).
//
//
package TutorialApps;

import org.omg.CORBA.*;
import SessionManager.*;
import Tutorial.*;  // Package for EAServer stub classes

public class JAConsole {

    static public void main(String options[]) {

        String _usage = "Usage: JAConsole iiop://<host>:<port>\n";
        String _ior = null;

       try {

           if (options.length >= 1)
           {
               _ior = options[0];
           }
           else
           {
                System.out.println(_usage);
                return;
           }

           //
           // Initialize the CORBA client-side ORB and
           // obtain a stub for the EAServer component instance.
           //
           System.out.println("... Creating session.");

           //
           // Initialize the ORB.
           //
           java.util.Properties props = new java.util.Properties();
           props.put("org.omg.CORBA.ORBClass", "com.sybase.CORBA.ORB");

           ORB orb = ORB.init(options, props);

           //
```

```
            // Create an instance of the EAServer SessionManager::Manager
            // CORBA IDL object.
            //

            Manager manager = ManagerHelper.narrow(orb.string_to_object(_ior))
;

            //
            // Create an authenticated session with user "Guest" and password
            // "GuestPassword".
            //
            Session session = manager.createSession("Guest", "GuestPassword");

            System.out.println("... Creating component instance.");

            //
            // Create a stub object instance for the
            // Tutorial/JavaArithmetic EAServer component.
            //
            Tutorial.JavaArithmetic comp =
            Tutorial.JavaArithmeticHelper.narrow(
            session.create("Tutorial/JavaArithmetic"));

            if (comp == null)
            {
              System.out.print("ERROR: Null component instance. ");
              System.out.print(
          "Check Jaguar Manager and verify that the component ");
              System.out.print(
          "Tutorial/JavaArithmetic exists and that it implements the ");
              System.out.println(
          "Tutorial::JavaArithmetic IDL interface.");
              return;
            }

            System.out.println("... Created component instance.");

             //
             // Invoke the multiply method.
             //
            System.out.println("... Multiplying:\n");
            double m1 = 3.1;
            double m2 = 2.5;
            double result = comp.multiply(m1, m2);
            System.out.println("    " + m1 + "*" + m2 + "=" + result);
```

```
        // Explicitly catch exceptions that can occur due to user error,
        // and print a generic error message for any other CORBA system
        // exception.

       } catch ( org.omg.CORBA.COMM_FAILURE cfe)
       {
           // The server is not running, or the specified URL is
           // wrong.
             System.out.println(
              "Error: could not connect to server at " + _ior + "\n"
             + "Make sure the specified address is correct and the "
             + "server is running.\n\n" + _usage );
       } catch ( org.omg.CORBA.OBJECT_NOT_EXIST cone )
        {
            // Requested object (component) does not exist.
            System.out.println(
               "Error: CORBA OBJECT_NOT_EXIST exception. Check the "
             + "server log file for more information. Also verify "
             + "that the Tutorial/JavaArithmetic "
             + "component has been created properly in "
            + "Jaguar Manager. \n");

        } catch (org.omg.CORBA.NO_PERMISSION npe) {
           // Login failed, or the component requires an authorization role
           // that this user is not a member of.
            System.out.println(
               "Error: CORBA NO_PERMISSION exception. Check whether "
             + "login authentication is enabled for your server and "
             + "whether the component has restricted access.\n");
             npe.printStackTrace();

       } catch (org.omg.CORBA.SystemException se)
       {
            // Generic CORBA exception
           System.out.println(
              "Received CORBA system exception: "
              + se.toString() );
           se.printStackTrace();
       }

        return;
    } // main()

}
```

Compile the application source using a JDK 1.3 or later compiler, for example:

```
%JAGUAR%\bin\jc JAConsole.java
```

Run the client program

If you have not refreshed or restarted the server since creating the JavaArithmetic component, refresh the server before running the client program.

Create a batch file or UNIX shell script to run the client application, then run it. The batch file or shell script configures the CLASSPATH environment variable, then runs the application using the JDK 1.3 java program included with your EAServer installation.

If necessary, you can run the client on a different machine than the server host, as long as your server uses a real host address and not localhost or 127.0.0.1.

❖ **Creating the Windows batch file**

- Create a file named *runja.bat* containing the commands below:

```
call %JAGUAR%\bin\setenv.bat
set CLASSPATH=%JAGUAR%\java\lib\easj2ee.jar;
set CLASSPATH=%CLASSPATH%;%JAGUAR%\java\lib\easclient.jar
set CLASSPATH=%CLASSPATH%;%JAGUAR%\html\classes
set JAVA_HOME=%JAGUAR_JDK13%
%JAVA_HOME%\jre\bin\java TutorialApps.JAConsole %*
```

❖ **Creating the UNIX shell script**

1 Create a file named *runja* containing the commands below:

```
#!/bin/sh
. $JAGUAR/bin/setenv.sh
CLASSPATH=$JAGUAR/java/lib/easj2ee.jar
CLASSPATH=$CLASSPATH:$JAGUAR/java/lib/easclient.jar
CLASSPATH=$CLASSPATH:$JAGUAR/html/classes export CLASSPATH
JAVA_HOME=$JAGUAR_JDK13
$JAVA_HOME/jre/bin/java TutorialApps.JAConsole $*
```

2 Change the file permissions to allow the script to be executed. For example:

```
chmod 777 runja
```

❖ **Running the client application**

- Run the batch or script file, specifying the server host name and IIOP port number on the command line as follows:

  ```
  runja iiop://host:iiop-port
  ```

  For example:

  ```
  runja iiop://myhost:9000
  ```

  If everything is working, the application prints the results from the invocation of the multiply method. If not, check the error text printed on the console where you ran the client, and check for error messages in the server log file.

Creating an applet client

These optional steps describe how to create an applet client that runs in a Web browser. Complete all steps under “Creating the application” on page 10 before running the applet:

1 Verify that your browser can run JDK 1.2 applets.

2 Prepare your EAServer installation to run applets.

3 Write the applet code.

4 Create an HTML page to run the applet.

5 Run the applet.

Verify that your browser can run JDK 1.2 applets

The EAServer client runtime requires JDK 1.2 or later.

To run the tutorial applet, you must have Web browser that is compatible with JDK 1.2 or a later JDK version. Most current browsers include a JDK 1.1 virtual machine. To run JDK 1.2 applets in these browsers, you must use the Java Plug-in available from Sun Microsystems at the Java Plug-in Web site at http://java.sun.com/products/plugin/.

The tutorial was tested with Netscape Navigator 4.6.*x*, 4.7.*x*, and Internet Explorer 5.0, all using Sun’s Java Plug-in, version 1.3.

The Java Plug-in is required
To run this tutorial without the Java Plug-in, you must edit the HTML code that loads the applet. The tutorial HTML code will not run the applet in browsers where the plug-in is not installed. The Java Plug-in may not be available on your UNIX platform; in that case use a browser running on a supported platform such as Windows NT, Windows 2000, or Sun Solaris.

If you are running the sample applet on a machine where EAServer has been installed, you may need to configure your browser to eliminate conflicts between EAServer classes that are downloaded with the applet and classes that are loaded from the local CLASSPATH setting. The simplest way to do this is to run the browser from a batch file or UNIX script as described here.

❖ **Configuring your browser on Windows**

This procedure is necessary only if you have EAServer classes in the system or user CLASSPATH environment variable, or you start the browser from a command line where CLASSPATH has been set to include EAServer classes.

1 Locate the browser executable, and create a batch file in the same directory with the same base file name, for example *netscape.bat* or *iexplore.bat*.

2 Add these commands to the batch file, where *browser.exe* is the name of the browser executable:

```
@ECHO OFF
SETLOCAL
SET CLASSPATH=""
START browser.exe %*
ENDLOCAL
```

3 Change shortcuts that run the browser so that they launch the batch file instead of the browser executable.

❖ **Configuring your browser on UNIX**

This procedure is necessary only if you have EAServer classes in the CLASSPATH when you start the browser process. For example, you may have modified the CLASSPATH setting in your *.login* or *.cshrc* file.

1 Create a script to run the browser, for example, *~/bin/netscape*.

2 Edit the script to set the CLASSPATH environment variable to an empty string before running the browser. For example (this script also sets the NPX_PLUGIN_PATH variable to configure the Sun Java Plug-in. See the Java Plug-in documentation for more information):

```
#!/bin/sh
set CLASSPATH="" export CLASSPATH
set NPX_PLUGIN_PATH=/opt/jre/1.3/plugin/sparc export NPX_PLUGIN_PATH
/opt/netscape/bin/netscape $*
```

3 Run this script to start your browser.

Prepare your EAServer installation to run applets

As installed, EAServer does not provide the classes required to run applets under the *html* or *html/classes* directories. To run the applet, you must copy or link the required runtime JAR files to the EAServer *html/classes* directory.

Do not expand the JAR files
The applets created in this tutorial do not require expansion of the *easclient.jar* and *easj2ee.jar* files.

❖ **On Windows, copy the JAR files**

- Copy *easclient.jar* and *easj2ee.jar* from the EAServer *java/lib* directory to the *html/classes* directory.

❖ **On UNIX platforms, link the JAR files**

- Create symbolic links from the *easclient.jar* and *easj2ee.jar* in the EAServer *java/lib* directory to the *html/classes* directory. For example, run these commands:

```
cd $JAGUAR/html/classes
ln -s $JAGUAR/java/lib/easclient.jar
ln -s $JAGUAR/java/lib/easj2ee.jar
```

Write the applet code

In the *html\classes* subdirectory of your EAServer installation, create a new directory called *TutorialApps* if it does not exist. In this directory, create the Java file below as *JAClient.java*.

You can find a copy of *JAClient.java* in the *html/docs/tutorial/java-corba* subdirectory of your EAServer installation. Here is the source for *JAClient.java*:

```
//
// This is a sample client applet that invokes the
// JavaArithmetic component created in the EAServer
// Java component tutorial.
//
package TutorialApps;

import org.omg.CORBA.*;
import SessionManager.*;
import java.awt.*;

import Tutorial.*;  // Package for EAServer stub classes

public class JAClient extends java.applet.Applet {

    // User interface controls
    Button mult_button;
    TextField m1_text;
    TextField m2_text;
    TextField  result_text;

    // Component stub instance
    Tutorial.JavaArithmetic _comp = null;


    public void init()
    {

        // Draw GUI controls
        m1_text = new TextField("      ");
        m1_text.setText("2.5");
        this.add(m1_text);
        Label l1 = new Label("*");
        this.add(l1);
        m2_text = new TextField("      ");
        m2_text.setText("3.1");
        this.add(m2_text);
        Label l2 = new Label("=");
        this.add(l2);
        result_text = new TextField("      ");
        result_text.setEditable(false);
        this.add(result_text);
        mult_button = new Button("Multiply");
        this.add(mult_button);
```

```
    //
    // Initialize the CORBA client-side ORB and
    // obtain a stub for the EAServer component instance.
    //
    System.out.println("... Creating session.");

    //
    // Initialize the ORB.
    //
    java.util.Properties props = new java.util.Properties();
    props.put("org.omg.CORBA.ORBClass", "com.sybase.CORBA.ORB");
    ORB orb = ORB.init(this, props);

    //
    // Create an instance of the EAServer SessionManager::Manager
    // CORBA IDL object.
    //
    String ior = this.getParameter("ior");
    Manager manager = ManagerHelper.narrow(
                      orb.string_to_object(ior));

    //
    // Create an authenticated session with user "Guest" and password
    // "GuestPassword".
    //
    Session session = manager.createSession("Guest", "GuestPassword");

    System.out.println("... Creating component instance.");

    //
    // Create a stub object instance for the
    // Tutorial/JavaArithmetic EAServer component.
    //
    _comp = Tutorial.JavaArithmeticHelper.narrow(
                session.create("Tutorial/JavaArithmetic"));

    if (_comp == null)
    {
      System.out.print("ERROR: Null component instance. ");
      System.out.print(
"Check Jaguar Manager and verify that the component ");
      System.out.print(
"Tutorial/JavaArithmetic exists and that it implements the ");
      System.out.println(
"Tutorial::JavaArithmetic IDL interface.");
      return;
```

```
        }

        System.out.println("... Created component instance.");

    } // init()

    // Handle button clicks
    public boolean action(Event e, java.lang.Object arg) {
        if (e.target == mult_button) {
            doMultiply();
            return true;
        }
        else return false;
    }

    // Call the multiply method and update the displayed result
    private void doMultiply() {

        // Harvest user input
        Double d1 = null;
        Double d2 = null;
        try {
            d1 = new Double(m1_text.getText());
            d2 = new Double(m2_text.getText());
        } catch (NumberFormatException nfe)
        {
            this.showStatus("ERROR: Bad number format.");
            result_text.setText("");
            return;
        }

        // Call the server component method
        double m1 = d1.doubleValue();
        double m2 = d2.doubleValue();
        try {
            double result = _comp.multiply(m1, m2);
            result_text.setText((new Double(result)).toString());
        }
        catch (org.omg.CORBA.SystemException se)
        {
            System.out.println(
               "Exception executing multiply method: "
               + se.toString() );
            se.printStackTrace();
        }
```

```
    } // doMultiply

}
```

Compile the applet source using a JDK 1.2 or later compiler, for example:

```
%JAGUAR%\bin\jc JAClient.java
```

Create an HTML page to run the applet

Copy the file *html/docs/tutorial/java-corba/runjavatut.html* to the *html/classes/TutorialApps* subdirectory in your EAServer installation. This file has the OBJECT and EMBED tags required to launch the applet in the Sun Java Plug-in:

```
<html><body bgcolor="#FFFFFF">
<head><title>This Applet runs the
EAServer tutorial Java component.</head></title>
<hr>
<center>

<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
WIDTH = 600 HEIGHT = 400  codebase="http://java.sun.com/products/plugin/1.3/j
install-13-win32.cab#Version=1,3,0,0">
<PARAM NAME = CODE VALUE = "TutorialApps/JAClient.class" >
<PARAM NAME = CODEBASE VALUE = "/classes" >
<PARAM NAME = ARCHIVE VALUE = "easj2ee.jar,easclient.jar" >
<PARAM NAME="type" VALUE="application/x-java-applet;version=1.3">
<PARAM NAME="scriptable" VALUE="false">
<PARAM NAME="ior" VALUE ="iiop://:9000">
<COMMENT>
<EMBED type="application/x-java-applet;version=1.3"
   CODE = "TutorialApps/JAClient.class"
   CODEBASE = "/classes"
   ARCHIVE = "easj2ee.jar,easclient.jar"
   WIDTH = 600 HEIGHT = 400
   ior = "iiop://:9000"
   scriptable=false
   pluginspage="http://java.sun.com/products/plugin/1.3/plugin-install.html">
<NOEMBED>
</COMMENT>
<h2>This would be a Cool Applet, but you are not running a Java enabled brows
er...</h2>
</NOEMBED>
</EMBED>
</OBJECT>
```

```
</center>
<hr>
</body></html>
```

No changes are required to this HTML file if your server is configured to use the default IIOP port number, 9000. If you (or your administrator) changed the IIOP port number after installing EAServer, edit the port number in the HTML file to match. You must edit the port number in both the OBJECT and the EMBED tag. In the OBJECT tag, the port number appears in this line:

```
<PARAM NAME = "ior" VALUE ="iiop://:9000">
```

In the EMBED tag, the port number appears in this line:

```
ior = "iiop://:9000"
```

About the OBJECT and EMBED tags

You can skip this section if your only goal is to run the tutorial applet. If you are modifying the tutorial code to run your own applets, you need to understand these tags.

The OBJECT tag is required to load the applet in Internet Explorer. The EMBED tag is required to load the applet in Netscape. You can run Sun's HtmlConverter tool to convert standard APPLET tags to equivalent OBJECT and EMBED tags. HtmlConverter is available at the Java Plug-in Web site at http://java.sun.com/products/plugin/.

The OBJECT and EMBED tags contain similar parameters. See the Java Plug-in Web site at http://java.sun.com/products/plugin/ for full documentation of the tag formats. The parameters of interest for this tutorial are:

- `CODE`, which specifies the applet class, `TutorialApps/JAClient.class`.
- `CODEBASE`, which specifies the download URL for the applet class and all other required classes. In this case, the classes are downloaded from the *classes* directory under the server's document root.
- `ARCHIVE`, which specifies JAR files required to run the applet. In this case, *easj2ee.jar* and *easclient.jar*. These files must exist at the location specified by the `CODEBASE` parameter. The JAR files must be specified in this order, because classes in *easclient.jar* depend on classes in *easj2ee.jar*.

  You can also load all class files from the code base, or all classes from a single JAR file. For best performance, applets should be deployed with all required classes in one JAR file. However, optimization is beyond the scope of this tutorial.

- `ior`, which is read by the applet source code to establish the IIOP connection to EAServer. The value is an IIOP URL of the form:

  `iiop://`*`host`*`:`*`iiop-port`*

  The host name is optional. The value `iiop://:9000` specifies port 9000, on the server from which the applet was downloaded.

Warning! If you specify a host name other than the Web server host name, users cannot run your applet unless they modify the security constraints in their installation of the Java Plug-in.

Run the applet

If you have not restarted or refreshed the server since creating the JavaArithmetic component, refresh the server before running the applet.

❖ **Running the applet**

1 Start your Web browser. If necessary, you can run the applet on a different machine than the server host, as long as your server uses a real host address and not localhost or 127.0.0.1.

2 Connect to the following URL, substituting your server's host name for *host*:

`http://`*`host`*`:8080/classes/TutorialApps/runjavatut.html`

As installed, EAServer uses 8080 as the HTTP port number. If your server uses a different HTTP port number, change the port number in the URL to match.

3 Enter numeric values to be multiplied, then click the Multiply button to invoke the multiply component method. The return value is displayed.

Debugging

If everything is working, you should be able to download the applet, watch as it is drawn, then invoke the multiply method by clicking the applet's Multiply button. If not, check the browser's Java console for error messages. Also check the server log file (*Jaguar.log* in the EAServer *bin* subdirectory).

If the applet connects to the server and instantiates the component successfully, you see `. . . created session` and `. . . created component instance` in the browser's Java console.

CHAPTER 3

Creating C++ Components and Clients

In this tutorial, you will create a C++ component, install it in EAServer, and create a C++ client program that connects to EAServer and calls a method in the component.

Topic	Page
Overview of the sample application	29
Tutorial requirements	30
Creating the application	30

For more information

For complete information on creating C++ components and C++ clients, see these chapters in the *EAServer Programmer's Guide*:

- Chapter 13, "CORBA C++ Overview"
- Chapter 14, "Creating CORBA C++ Components"
- Chapter 15, "Creating CORBA C++ Clients"

Overview of the sample application

In this sample:

1 The client-side executable, developed with C++, instantiates the middle-tier C++ component, CPPArithmetic.

2 The client executable calls the multiply method in CPPArithmetic.

3 The multiply method computes the product of the input values, then returns the result.

4 The client executable displays the result for the end user.

Tutorial requirements

To create this tutorial application, you need:

- The EAServer software. The *EAServer Installation Guide* for your platform describes how to install the software.
- A C++ development environment, such as:
  - For Windows, Microsoft Visual C++.
  - For UNIX, the system C++ compiler. Some UNIX platforms support multiple C++ versions. The *EAServer Release Bulletin* for your UNIX platform lists compilers that have been tested with EAServer.

Creating the application

To create and run the sample application:

1 Verify your environment.

2 Start EAServer and EAServer Manager.

3 Define a package, component, and method.

4 Generate stubs, skeletons, and implementation templates.

5 Write the server-side code.

6 Write the client-side code.

7 Compile the client executable.

8 Run the client executable.

Verify your environment

Before running the tutorial, verify these environment settings:

- For all platforms, the JAGUAR environment variable must be set to the location of your EAServer installation.
- For Windows, the PATH environment variable must include the EAServer *dll* subdirectory.

- For UNIX platforms, the EAServer *lib* directory must be added to the shared library search path variable listed in Table 3-1 for your platform. If running on Solaris, and you use the Solaris version 4.*x* compiler, the EAServer *lib_sol4x* directory must be in LD_LIBRARY_PATH.

Table 3-1: Shared library search path variables for UNIX platforms

Platform	Variable name
Solaris	LD_LIBRARY_PATH
HP-UX	SHLIB_PATH
AIX	LIBPATH
Linux	LD_LIBRARY_PATH

❖ **Configuring the Windows environment**

- To configure the command line where you are running the tutorials, run these commands, substituting your EAServer installation location for *eas-home*:

```
set JAGUAR=eas-home
set PATH=%JAGUAR%\dll;%PATH%
```

You can also edit these variables in the System dialog for the Windows Control Panel, or create a batch file to configure the settings.

❖ **Configuring the UNIX environment for C shell**

- To configure the C shell session where you are running the tutorials, run these commands, substituting your EAServer installation location for *eas-home*, and the shared-library variable from Table 3-1 for *LIB_PATH*:

```
setenv JAGUAR eas-home
setenv LIB_PATH $JAGUAR/lib:$LIB_PATH
```

If running on Solaris, using a version 4.*x* CC compiler:

```
setenv JAGUAR eas-home
setenv LD_LIBRARY_PATH \
$JAGUAR/lib_sol4x:$LD_LIBRARY_PATH
```

❖ **Configuring the UNIX environment for Bourne shell**

- To configure the Bourne shell session where you are running the tutorials, run these commands, substituting your EAServer installation location for *eas-home*, and the shared-library variable from Table 3-1 for *LIB_PATH*:

```
JAGUAR=eas-home export JAGUAR
LIB_PATH=$JAGUAR/lib:$LIB_PATH export LIB_PATH
```

If running on Solaris, using a version 4.*x* CC compiler:

```
JAGUAR=eas-home export JAGUAR
LD_LIBRARY_PATH=$JAGUAR/lib_sol4x:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

Start EAServer and EAServer Manager

❖ **Starting the server**

- If the server is not already running, follow the instructions under "Starting the server" on page 3.

❖ **Starting EAServer Manager**

- If EAServer Manager is not already running, start it as described in "Using EAServer Manager" on page 4.

Define a package, component, and method

This section shows you how to use EAServer Manager to create the package, component, and method for the sample application.

For complete information on configuring packages, components, and methods, see Chapter 5, "Defining Component Interfaces," in the *EAServer Programmer's Guide*.

Define a new package

In EAServer, a package is a unit of deployment for a group of components that perform related tasks. All components created in the EAServer tutorials are installed in the Tutorial package.

Before a component can be instantiated by clients, it must be installed in a package, and that package must be installed in the server. The steps below create the package and component within the predefined "Jaguar" server to satisfy these requirements.

❖ **Creating the *Tutorial* package if it does not exist**

1 In EAServer Manager, expand the servers folder, then expand the Jaguar server icon.

2 Expand the Installed Packages folder. If the Tutorial package is displayed, skip to "Define and install a new component" on page 33.

3 Highlight the Installed Packages folder, and select File | Install Package.

In the Package wizard, select Create and Install a New Package.

For the package name, enter `Tutorial`.

4 Click Create New Package.

You see the Package Properties window.

5 Click OK.

Define and install a new component

You will define a new C++ component, CPPArithmetic.

❖ Defining the component

1 Click on the Tutorial package.

2 Select File | New Component.

3 In the New Component wizard, select Define New Component.

For the component name, enter `CPPArithmetic`.

4 Click Finish.

You see the Component Properties window.

5 Select the General tab. Fill in the fields as follows:

Field	Value
Description	Tutorial C++ component
Codeset	Server's Codeset (default)
Component Type	C++
DLL Name	`libCPPArithmetic` (no extension)
C++ Class	`CPPArithmeticImpl`
C++ Executable	(blank)
Use Platform Independent Naming	Unchecked

6 Leave the remaining fields at their default settings.

7 Click OK.

Define the multiply method

The component interface will have one method, multiply.

❖ **Defining the component interface**

1. Expand the Tutorial package. Double-click the CPPArithmetic component to show the Roles and Interfaces folders beneath it.
2. Double-click the Interfaces folder, and highlight the Tutorial::CPPArithmetic interface. If you do not see this interface, install it as follows:
   a. Highlight the Interfaces folder and select File | Add Interfaces ...
   b. In the Install Interface dialog box, highlight Tutorial::CPPArithmetic in the Selected to Install table, then click Install.
   c. Highlight the Tutorial::CPPArithmetic interface that is now displayed under the Interfaces folder.
3. With the Tutorial::CPPArithmetic interface highlighted, select File | New Method.
4. Assign the name multiply to the method.
5. Click Create New Method.

   You see the Method Properties window.
6. In the Return field, select double as the method's return type.
7. Beneath the empty parameter list, click Add to add a parameter. In the New Parameter dialog:
   - For the parameter name, enter `m1`.
   - For Mode, select `in`.
   - For Type, select `double`.
8. Click OK to close the New Parameter dialog box.
9. Repeat steps 7 and 8 to add a second parameter named `m2` with a Type of double.
10. Click OK to close the Method Properties dialog box.

Generate stubs, skeletons, and implementation templates

Once you have created the package, component, and methods, you can generate the stub and skeleton files for the component. The client executable uses the stubs to invoke the server-side component methods. The server uses the skeleton to invoke your component implementation class.

❖ **Generating the stubs and skeletons**

1 Click on the Tutorial package and select the CPPArithmetic component.

2 Select File | Generate Stubs/Skeletons.

3 Deselect Generate Java Stubs.

4 Select Generate Stubs, then check C++ Stubs. Leave the C/C++ Code Base field at the default setting, which should be the full path to your EAServer *include* subdirectory.

5 Click Next to display the skeleton generation options, and select Generate Skeletons.

6 Under Skeletons, select Generate Skeletons on Client and leave the C/C++ Code Base field at the default setting, which should be the full path to your EAServer *cpplib* subdirectory.

7 Click Finish.

Write the server-side code

EAServer Manager has generated C++ implementation templates for the component methods. Here we will fill in the implementation template, then build a shared library or DLL file. Finally, we will verify that the shared library or DLL is in the EAServer *cpplib* subdirectory, where EAServer expects to find C and C++ component library files.

❖ **Writing the server-side code**

1 Navigate to the *cpplib* directory under your EAServer installation, then navigate to the *Tutorial/CPPArithmetic* subdirectory. You should see the following files:

- **CPPArithmeticImpl.hpp.new** Template for the component header file. Defines the CPPArithmeticImpl class. No changes are required for the tutorial, other than renaming the file as discussed below.

- **CPPArithmeticImpl.cpp.new** Template for the component implementation. Contains the definition of the component methods. Changes you must make to this file are described below.
- **Tutorial_CPPArithmetic.cpp** Source for the skeleton. Do not modify the generated skeleton code.
- **make.nt** Microsoft nmake makefile. The nmake utility is included with the Microsoft Visual C++ installation.
- **make.unix** UNIX makefile, for all UNIX platforms.

2 Rename the implementation files to *CPPArithmeticImpl.hpp* and *CPPArithmeticImpl.cpp*. (In other words, remove the *.new* extension from both file names).

3 Open *CPPArithmeticImpl.cpp* in a text editor, then find the definition of the multiply method. Change the definition so that it matches the one below:

```
CORBA::Double CPPArithmeticImpl::multiply
    (CORBA::Double m1,
    CORBA::Double m2)
{
    CORBA::Double result;
    result = m1 * m2;
    return result;
}
```

4 Save your changes.

❖ **Building the component on Windows**

1 Verify your setup as described in “Verify your environment” on page 30.

2 Rename *make.nt* to *Makefile*, then open *Makefile* in a text editor. Find the definition of the ODBCHOME macro:

```
ODBCHOME=d:\msdev
```

3 Change the ODBCHOME definition to match the directory where you have installed Microsoft Visual C++, for example:

```
ODBCHOME="D:\engapps\devStudio\VC98"
```

4 Save your changes.

5 Build the DLL by running nmake (no arguments are required).

You should see a new file called *libCPPArithmetic.dll*. Verify that the makefile has copied this file to the EAServer *cpplib* subdirectory. If nmake fails, verify that you have renamed the *.cpp* and *.hpp* implementation files with the expected file names, and that you have applied the correct edits to *CPPArithmeticImpl.cpp* and *Makefile*.

❖ **Building the component on UNIX platforms**

1 Verify your setup as described in "Verify your environment" on page 30.

2 Rename *make.unix* to *Makefile*.

3 On Solaris, the make file is compatible with the Solaris CC compiler, version 6.*x*. If you use a version 4.*x* compiler, edit *Makefile* and change `-L$JAGUAR/lib` to `-L$JAGUAR/lib_sol4x`. You must use the compiler version that is compatible with your server. The default server binary requires libraries compatible with the 6.*x* compiler, but you can override this setting when starting the server. For more information, see "Starting the server" in the *EAServer System Administration Guide*.

4 Build the shared library by running make (no arguments are required).

You should see a new file called *libCPPArithmetic.ext*, where *ext* is the appropriate shared library extension for your platform. Verify that the makefile has copied this file to the EAServer *cpplib* subdirectory.

If make fails, verify the following:

- You have renamed the *.cpp* and *.hpp* implementation files with the expected file names, and that you have applied the correct edits to *CPPArithmeticImpl.cpp*.
- The compile and link settings in *Makefile* are appropriate for your installation. The settings are defined in the file *cpplib/make.include.plat*, where *plat* is the platform code returned by running `uname -s` on your system. If necessary, edit this file to match your system configuration.

Write the client-side code

Create the source file for the sample C++ client, *arith.cpp*. You can find a copy of *arith.cpp* in the *html/docs/tutorial/cpp* subdirectory of your EAServer installation. Here is the source for *arith.cpp*:

```
/*
** arith.cpp -- Example C++ client for the EAServer C++
**   tutorial.
```

```
**
**   This program connects to EAServer,
**   creates an instance of the Tutorial/CPPArithmetic
**   component, and invokes the multiply method.
**
** Usage:
**    arith iiop://<host>:<port>
**
**    Where:
**
**       <host> is the host name or IP address of the server machine.
**
**       <iiop-port> is the server's IIOP port (9000 in the
**         default configuration).
**
*/

#include <stdio.h>
#include <iostream.h>
#include <string.h>
#include <Jaguar.hpp>
#include <SessionManager.hpp>
#include <Tutorial.hpp>    // Stubs for interfaces in Tutorial IDL
                           // module.

int main(int argc, char** argv)
{
    const char *usage =
"Usage:\n\tarith iiop://<host>:<iiop-port>\n";
    const char *tutorial_help =
"Check Jaguar Manager and verify that the"
"Tutorial/CPPArithmetic component exists "
"and that it implements the "
"Tutorial::CPPArithmetic IDL interface.";

    const char *component_name = "Tutorial/CPPArithmetic";

    try {

        if (argc < 2)
        {
          cout << usage;
          return -1;
        }

        char* manager_url = argv[1];
```

```
 cout << "**** Creating session\n";

 // Initialize the ORB
 CORBA::ORB_var orb = CORBA::ORB_init(argc, argv, 0);

// Create a SessionManager::Manager instance

CORBA::Object_var obj =
    orb->string_to_object(manager_url);
SessionManager::Manager_var manager =
  SessionManager::Manager::_narrow(obj);
 if (CORBA::is_nil(manager))
 {
    cout << "Error: Null SessionManager::Manager instance. Exiting. "
         << usage ;
    return -1;
 }

 // Create an authenticated session for user Guest
 // using password GuestPassword

 SessionManager::Session_var session =
   manager->createSession("Guest", "GuestPassword");
 if (CORBA::is_nil(session))
 {
   cout << "Error: Null session. Exiting. " << usage;
   return -1;
 }

 // Obtain a factory for component instances by
 // resolving the component name

 cout << "**** Creating component instance for "
      << component_name << "\n" ;

 obj = session->lookup(component_name);
 SessionManager::Factory_var arithFactory =
 SessionManager::Factory::_narrow(obj);

 if (CORBA::is_nil(arithFactory))
 {
    cout << "ERROR: Null component factory for component "
         << component_name
         << tutorial_help ;
     return -1;
```

```
    }

   // Use the factory to create an instance.

    Tutorial::CPPArithmetic_var arith =
       Tutorial::CPPArithmetic::_narrow(arithFactory->create());

    // Verify that we really have an instance.
    if (CORBA::is_nil(arith)) {
        cout << "ERROR: Null component instance. "
             << tutorial_help ;
        return -1;
    }


    // Call the multiply method.

    cout << "**** Multiplying ...\n\n";
    CORBA::Double m1 = (CORBA::Double)3.1;
    CORBA::Double m2 = (CORBA::Double)2.5;
    CORBA::Double result = arith->multiply(m1, m2);

    cout << (double)m1 << " * " << (double)m2
         << " = " << (double)result
         << "\n\n";

}

// Explicitly catch exceptions that can occur due to user error,
// and print a generic error message for any other CORBA system
// exception.

// Requested object (component) does not exist.
catch ( CORBA::OBJECT_NOT_EXIST cone )
{
    cout << "Error: CORBA OBJECT_NOT_EXIST exception. Check the "
         << "server log file for more information. Also verify "
         << "that the " << component_name
         << " component has been created properly in "
         << "Jaguar Manager. " << tutorial_help ;
}

// Authentication or authorization failure.
catch ( CORBA::NO_PERMISSION npe )
{
    cout << "Error: CORBA:: NO_PERMISSION exception. Check whether "
```

```
            << "login authentication is enabled for your server and "
            << "whether the component has restricted access. If so "
            << "edit the source file to use a valid user name and "
            << "password.\n";
    }

    // Invalid object reference.
    catch ( CORBA::INV_OBJREF cio )
    {
        cout << "Error: CORBA INV_OBJREF exception.";
    }

    // Communication failure. Server could be down or URL's port value
    // could be wrong.
    catch ( CORBA::COMM_FAILURE ccf )
    {
        cout << "Error: CORBA COMM_FAILURE exception. Check that the "
             << "specified host and IIOP port number are "
             << "correct and that the server is running. "
             << usage;
    }

    // Anything else.
    catch ( CORBA::OBJ_ADAPTER )
    {
        cout << "Error: CORBA::OBJ_ADAPTER \n";
    }
    catch ( CORBA::SystemException cse )
    {
        cout << "Error: CORBA System Exception. Check that the server "
             << "hostname and IIOP port are specified correctly, and "
             << "check the server's error log for more information.\n"
             << usage;
    }

    return 0;
}
```

Compile the client executable

❖ **Compiling the client on Windows**

1 Verify your setup as described in “Verify your environment” on page 30.

2 Create a batch file with these commands and run it:

```
SETLOCAL
call %JAGUAR%\bin\setenv.bat
set INCLUDE=.;%JAGUAR%\include;%INCLUDE%;
set INCLUDE=%INCLUDE%;%JAGUAR_JDK13%\include;
set INCLUDE=%INCLUDE%;%JAGUAR_JDK13%\include\win32
set LIB=%JAGUAR%\lib;%LIB%
cl /W3 /nologo /DWIN32 /Gd /GX -c arith.cpp
set SYSLIBS=kernel32.lib advapi32.lib
link /MAP /out:arith.exe arith.obj libjcc.lib %SYSLIBS%
ENDLOCAL
```

❖ **Compiling the client on UNIX**

1. Verify your setup as described in "Verify your environment" on page 30.
2. Create a shell script containing the commands for your platform from Table 3-2, then run the shell script.
3. Change the script file permissions to allow execution, for example, assuming you have named the script *compile.sh*:

```
chmod 777 compile.sh
```

Table 3-2: Client compilation commands for UNIX platforms

Platform	Shell script

Solaris	This shell script works with the Solaris CC compiler, version 6.*x*:
	`#!/bin/sh`
	`. $JAGUAR/bin/setenv.sh`
	`CC -DJAG_NO_NAMESPACE -z muldefs -I. -I$JAGUAR/include \`
	`-I$JAGUAR_JDK13/include -I$JAGUAR_JDK13/include/solaris \`
	`-L$JAGUAR/lib -ljcc -ljtml_r -ljtli_r -ljlog -lunic -l nsl \`
	`-l dl -l thread -l m -o arith arith.cpp`
	If you use the version 4.*x* compiler, change `-L$JAGUAR/lib` to `-L$JAGUAR/lib_sol4x`.

HP-UX	This shell script uses the HP-UX ANSI C++ (aCC) compiler:
	`#!/bin/sh`
	`. $JAGUAR/bin/setenv.sh`
	`aCC -c +DA1.1 +DS2.0 +u4 -DNATIVE -D_HPUX -D_POSIX_C_SOURCE=199506L \`
	`-D_HPUX_SOURCE -I $(JAGUAR_JDK13)/include -I`
	`$(JAGUAR_JDK13)/include/hp-ux \`
	`-I $(JAGUAR_JDK12)/include -I $(JAGUAR_JDK12)/include/hp-ux -I. \`
	`-I$JAGUAR/include -L$JAGUAR/lib -lpthread -ljcc -lnsl -ljtml_r \`
	`-ljlog -ljinsck_r -lunic -o arith arith.cpp`

Platform	Shell script

AIX	This shell script uses the IBM native compiler:
	`#!/bin/sh`
	`. $JAGUAR/bin/setenv.sh`
	`xlC_r -g -c -DDEBUG -DJAG_NO_NAMESPACE -DAIX -D_AIX -qcpluscmt -qnoro \`
	`-qmaxmem=-1 -qarch=com -qtbtable=full -I $(JAGUAR_JDK13)/include \`
	`-I $(JAGUAR_JDK12)/include -I. -I$JAGUAR/include \`
	`-brtl -L$JAGUAR/lib -ljcc.so -lunic -ljtml_r.so -ljinsck_r.so \`
	`-ljlog -lpthread -lnsl -o arith arith.cpp`

Linux	This shell script uses the g++ compiler:
	`#!/bin/sh`
	`. $JAGUAR/bin/setenv.sh`
	`g++ -c -D_GNU_SOURCE=1 -DLINUX -D_LINUX -D_REENTRANT -fPIC \`
	`-fwritable-strings -pipe -g -DDEBUG -I $(JAGUAR_JDK13)/include \`
	`-I $(JAGUAR_JDK13)/include/linux -I $(JAGUAR_JDK12)/include \`
	`-I $(JAGUAR_JDK12)/include/linux -I. -I$JAGUAR/include \`
	`-L$JAGUAR/lib -lpthread -ljcc -lnsl -ljtml_r -ljinsck_r \`
	`-ljlog -l unic -o arith arith.cpp`

Run the client executable

If you have not refreshed or restarted the server since creating the CPPArithmetic component, do so now before running the client program. Make sure your environment is configured as described in “Verify your environment” on page 30.

Run the executable, specifying the server host name and IIOP port number on the command line as follows:

```
arith iiop://host:iiop-port
```

For example:

```
arith iiop://myhost:9000
```

If everything is working, arith prints the results from the invocation of the multiply method. If not, check the error text printed on the console where you ran the client, and check for error messages in the server log file.

CHAPTER 4

Creating Enterprise JavaBeans Components and Clients

In this tutorial, you will create two Enterprise JavaBeans (EJB) components and a simple client to verify their operation.

Topic	Page
Overview of the sample components	45
Tutorial requirements	45
Creating the application	46

For more information

For complete information on creating EJB components and Java clients, see these chapters in the *EAServer Programmer's Guide*:

- Chapter 6, "Enterprise JavaBeans Overview"
- Chapter 7, "Creating Enterprise JavaBeans Components"
- Chapter 8, "Creating Enterprise JavaBeans Clients"

Overview of the sample components

The application consists of two components used to manage a glossary of terms and components, and a relational database that manages the glossary data. You will create an EJB entity bean that allows you to search for keyword entries, create and delete keyword entries, and modify definitions. You will also create an EJB stateless session bean that can be used to run arbitrary select queries against the database.

Tutorial requirements

To create the tutorial application, you need:

- The EAServer software

The *EAServer Installation Guide* for your platform describes how to install the software.

- Java development environment

  The tutorial steps use the JDK software that is included with your EAServer installation. You can also use JBuilder or any development tool that is compatible with JDK 1.3 or later.

  EAServer includes a JDK 1.3 installation, and scripts to use this JDK compiler. See "Java compiler scripts" on page 10 for more information.

- Adaptive Server Anywhere, version 7.0 or later

  The sample database requires Adaptive Server Anywhere, version 7.0 or later. This software is optionally installed with EAServer on Windows and supported UNIX platforms such as Solaris, HP-UX, Linux, and IBM AIX. See the *EAServer Installation Guide* for installation instructions.

  If you are using another platform, see the *EAServer Release Bulletin* for your platform.

Creating the application

To create and run the sample application:

1 Start EAServer and EAServer Manager.

2 Create the package and components.

3 Create the glossary database and connection cache.

4 Create the client application.

5 Run the client application.

Start EAServer and EAServer Manager

❖ **Starting EAServer**

- If EAServer is not already running, follow the instructions under "Starting the server" on page 3 to start the server.

❖ **Starting EAServer Manager**

- If EAServer Manager is not already running, start it as described in "Using EAServer Manager" on page 4.

Create the package and components

In EAServer, a package is a unit of deployment for a group of components that perform related tasks. Before a component can be instantiated by clients, it must be installed in a package, and that package must be installed in the server. The steps below create the package and component within the predefined "Jaguar" server to satisfy these requirements.

Define a new package

All components created in the EAServer tutorials are installed in the Tutorial package.

❖ **Creating the *Tutorial* package if it does not exist**

1 In EAServer Manager, expand the servers folder, then expand the Jaguar server icon.

2 Expand the Installed Packages folder. If the Tutorial package is displayed, skip to "Define the Glossary entity bean" on page 47.

3 Highlight the Installed Packages folder, and select File | Install Package.

In the Package wizard, select Create and Install a New Package.

For the package name, enter `Tutorial`.

4 Click Create New Package.

You see the Package Properties window.

5 Click OK.

Define the Glossary entity bean

We will define the entity bean by first creating the bean interface and implementation classes in Java, then importing the classes into EAServer Manager. Importing the classes creates the EJB component and defines the IDL interfaces required for it to run in Jaguar and be invoked by clients.

❖ **Creating the entity bean classes**

1 Under the EAServer *java/classes* directory, create the following subdirectory structure:

```
Sample/Intro/Glossary
```

2 Copy the following files from the *html/docs/tutorial/ejb* directory of your installation to the *java/classes/Sample/Intro/Glossary* directory:

- *Glossary.java* defines the remote interface.
- *GlossaryHome.java* defines the home interface.
- *GlossaryBean.java* contains the source for the implementation.

3 Compile these classes using a JDK 1.3 or later compiler, for example, on UNIX:

```
cd $JAGUAR/java/classes/Sample/Intro/Glossary
$JAGUAR/bin/jc Glossary*.java
```

Or on Windows, in a Command window:

```
cd %JAGUAR\java\classes\Sample\Intro\Glossary
%JAGUAR%\bin\jc Glossary*.java
```

❖ **Importing the entity bean classes into EAServer Manager**

1 In EAServer Manager, click on the Tutorial package.

2 Select File | New Component.

3 In the Component wizard, select Import From EJB Class File.

4 In the Component wizard - CLASSPATH screen, click Next. No changes are required to the default CLASSPATH to import our classes.

5 In the Component wizard - Import EJB Class Files screen, enter the values below:

Field	Value
Component Name	`Glossary`
Component Type	JaguarEJB::EntityBean
Bean Class	`Sample.Intro.Glossary.GlossaryBean`
Primary Key Class	`java.lang.String`
Specify Remote Interface	(Checked)
Home Interface Class	`Sample.Intro.Glossary.GlossaryHome`

Field	Value
Remote Interface Class	`Sample.Intro.Glossary.Glossary`
Specify Local Interfaces	(Not checked)

6 Click Finish. You see a dialog saying "All methods imported," then the Component Properties dialog box displays.

7 Apply or confirm the following settings to the General tab fields in the Component Properties dialog box:

Field	Value
Description	`Tutorial EJB entity bean`
Component Type	EJB - Entity Bean
EJB Version	2.0
JNDI Name	`Tutorial/Glossary`

Leave other fields as-is, and click OK.

Define the Query stateless session bean

As done to create the entity bean, we will define the session bean by first creating the bean interface and implementation classes in Java, then importing the classes into EAServer Manager.

❖ **Creating the session bean classes**

1 Copy the following files from the *html/docs/tutorial/ejb* directory of your installation to the *java/classes/Sample/Intro/Glossary* directory:

- *Query.java* defines the remote interface.
- *QueryHome.java* defines the home interface.
- *QueryBean.java* contains the source for the implementation.

2 Compile these classes using a JDK 1.3 or later compiler, for example, on UNIX:

```
cd $JAGUAR/java/classes/Sample/Intro/Glossary
$JAGUAR/bin/jc Query*.java
```

Or on Windows, in a Command window:

```
cd %JAGUAR\java\classes\Sample\Intro\Glossary
%JAGUAR%\bin\jc Query*.java
```

❖ **Importing the classes into EAServer Manager**

1 In EAServer Manager, click on the Tutorial package.

2 Select File | New Component.

3 In the Component wizard, select Import From EJB Class File.

4 In the Component wizard - CLASSPATH screen, click Next. No changes are required to the default CLASSPATH to import our classes.

5 In the Component wizard - Import EJB Class Files screen, enter the values below:

Field	Value
Component Name	`Query`
Component Type	JaguarEJB::StatelessSessionBean
Bean Class	`Sample.Intro.Glossary.QueryBean`
Specify Remote Interface	(Checked)
Home Interface Class	`Sample.Intro.Glossary.QueryHome`
Remote Interface Class	`Sample.Intro.Glossary.Query`
Specify Local Interfaces	(Not checked)

6 Click Finish. You see a dialog saying “All methods imported,” then the Component Properties dialog box displays.

7 Apply or confirm the following settings to the General tab fields in the Component Properties dialog box:

Field	Value
Description	`Tutorial EJB session bean`
Component Type	EJB - Stateless SessionBean
EJB Version	2.0
JNDI Name	`Tutorial/Query`

Leave other fields as-is, and click OK.

Generate stubs and skeletons

Use EAServer Manager to generate stubs and skeletons for the new components. The skeleton contains generated code to manage the interaction between EAServer and the implementation. The stubs are required by clients to execute the component.

❖ **Generating stubs and skeletons**

1 Click on the Tutorial package and select the Glossary component.

2 Select File | Generate Stub/Skeleton.

3 Deselect Generate Stubs.

4 Click Next to display the skeleton generation options and configure them as follows:

   a Select Generate Skeletons.

   b Select Generate Skeletons on Server.

   c Select Compile Java Skeletons.

5 Click Next to display the advanced options and configure them as follows:

   a For Java Version, select JDK 1.2 and Above.

   b For Generation Strategy, choose Full.

6 Click Finish.

7 Repeat these steps to generate stubs and skeletons for the `Query` component.

Explicit stub generation is not required
When generating skeletons, EAServer Manager generates stubs under the skeleton code base, *java/classes*. We will use these stubs to run our client.

Create the glossary database and connection cache

The glossary data is stored in an Adaptive Server Anywhere database. To make the data available to the components, we must start a database server, define a connection cache, and associate the connection cache with each component.

Start the glossary database server

Copy the file *gloss.db* from the *html/docs/tutorial/ejb/database* directory to the *sample* directory of your EAServer installation. Create a batch or script file to run the database server as described below.

❖ **Creating and running the Windows batch file**

1 In the *sample* directory of your EAServer installation, create a batch file named *run_gloss.bat*, containing the commands below. These commands start the database server on port 2640. If that port is in use on your machine, edit the port number to an unused value:

```
SETLOCAL
call %JAGUAR%\bin\setenv.bat
cd %JAGUAR%\sample
start dbsrv7 -x tcpip(ServerPort=2640) -n localhost gloss.db
ENDLOCAL
```

2 Start the database by running *run_gloss.bat*. For example, double-click this file in Windows Explorer.

❖ **Creating and running the UNIX script file**

1 Edit the *bin/setenv.sh* script in your EAServer installation, and verify that the SQLANY setting matches the location where you have installed Adaptive Server Anywhere.

2 Create a text file *run_gloss*, containing the commands below. These commands start the database server on port 2640. If that port is in use on your machine, edit the port number to an unused value:

```
#!/bin/sh
#
# If JAGUAR is not set where you run this, uncomment this
# line and edit the path to match your install location:
#JAGUAR=/path/to/your/install export JAGUAR
. $JAGUAR/bin/setenv.sh

cd $JAGUAR/sample
$SQLANY/bin/dbsrv7 -x "tcpip(ServerPort=2640)" -n localhost gloss.db
```

3 Change the file permissions to allow the script to be executed. For example:

```
chmod 777 run_gloss
```

4 Run the script in a terminal window.

Create the connection cache and associate it with the components

A connection cache maintains a pool of connections to a database server, increasing performance by allowing connection sharing and reuse. EJB connections obtain database connections using JNDI. In EAServer, we must associate a connection cache with the JNDI name alias that the component uses to look up connections.

❖ Creating the connection cache

1 Click on the Connection Caches folder in EAServer Manager, and select File | New Connection Cache.

2 Enter `Glossary` as the name.

3 In the Connection Cache Properties dialog, configure these settings:

Tab/ Setting	Value
General/ Description	`Glossary database`
General/ Server name	`NetworkProtocol=Tds:Server=`*localhost*`:Port=`*2640* **Port numbers must match** Make sure the port number used in the connection cache properties matches the port number you specify in the start script or batch file for the database server.
General/ User name	`dba`
General/ Password	`sql`
Driver/ DLL or Class Name	`com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource`
Driver/ Type	JDBC
Caching/ Enable Cache-by-Name Access	(Selected)

4 Leave other settings as-is and click OK.

5 Verify the connection cache properties as follows:

a Highlight the Glossary cache and select File | Properties.

b Click Refresh, then click Ping. If the Ping operation fails, confirm that you have applied the settings correctly and that the database is running.

❖ **Associating the cache with the EJB components**

1 Click on the Tutorial package and select the Glossary component.

2 Choose File | Component Properties.

3 In the Component Properties dialog box, display the Resource Refs tab.

4 Click Add to create a new reference in the list. Configure the values as follows:

a Set the Name field to `jdbc/glossary`.

b For Type, choose java.sql.Datasource.

c For Authentication, choose Container.

5 Click in the Resource Link field at the bottom of the dialog, and choose Glossary from the drop-down list.

6 Click OK to save the changes.

7 Repeat these steps for the Query component.

Create the client application

Copy the file *TestClient.java* from the *html/docs/tutorial/ejb* directory to the *java/classes/Sample/Intro/Glossary* directory of your EAServer installation.

Compile this file with a JDK 1.3 or later compiler, for example, on UNIX:

```
cd $JAGUAR/java/classes/Sample/Intro/Glossary
$JAGUAR/bin/jc TestClient.java
```

Or on Windows, in a Command window:

```
cd %JAGUAR\java\classes\Sample\Intro\Glossary
%JAGUAR%\bin\jc TestClient.java
```

Run the client application

If you have not refreshed or restarted your server since last modifying the Query or Glossary components, refresh the server now before running the client. Otherwise, verify that the server is running.

Run the client from using a batch file or UNIX shell script. The batch file or shell script configures the CLASSPATH environment variable, then runs the application using the JDK 1.3 java program included with your EAServer installation.

❖ **Creating the Windows batch file**

- Create a file named *runtest.bat* containing the commands below:

```
call %JAGUAR%\bin\setenv.bat
set CLASSPATH=%JAGUAR%\java\lib\easj2ee.jar;
set CLASSPATH=%CLASSPATH%;%JAGUAR%\java\lib\easclient.jar
set CLASSPATH=%CLASSPATH%;%JAGUAR%\java\classes
set JAVA_HOME=%JAGUAR_JDK13%
%JAVA_HOME%\jre\bin\java Sample.Intro.Glossary.TestClient %*
```

❖ **Creating the UNIX shell script**

1 Create a file named *runtest* containing the commands below:

```
#!/bin/sh
. $JAGUAR/bin/setenv.sh
CLASSPATH=$JAGUAR/java/lib/easj2ee.jar
CLASSPATH=$CLASSPATH:$JAGUAR/java/lib/easclient.jar
CLASSPATH=$CLASSPATH:$JAGUAR/java/classes export CLASSPATH
JAVA_HOME=$JAGUAR_JDK13
$JAVA_HOME/jre/bin/java Sample.Intro.Glossary.TestClient $*
```

2 Change the file permissions to allow the script to be executed. For example:

```
chmod 777 runtest
```

❖ **Running the client application**

- Run the batch or script file, specifying the server host name and IIOP port number on the command line as follows:

```
runtest iiop://host:iiop-port
```

For example:

```
runtest iiop://myhost:9000
```

The client application:

1 Creates a proxy for the Glossary entity bean's home interface, then calls the create method to populate the database with some glossary entries.

2 Creates a proxy for Query session bean's home interface, then calls the runQuery method to get a result set containing all the entries, then prints them.

If errors occur, check the server log file for information on how to correct the problem.

CHAPTER 5

Using the EAServer Samples

This chapter describes the samples included in the EAServer installation and where to find additional samples on the World Wide Web.

Topic	Page
Samples in the EAServer installation	57
Java Pet Store	61
PowerBuilder samples	62
Web Services Toolkit samples	62
Samples on the Sybase Web site	62

Samples in the EAServer installation

In addition to the CORBA Java and EJB tutorials in this book, the EAServer installation includes many optional Java and C++ samples.

Java samples

Your EAServer installation includes the following Java samples.

Using JavaServer Pages and EJB components

The MyPortfolio sample demonstrates the use of JavaServer Pages (JSPs) and EJB components. The application simulates an online stock trading service. A Web application runs JSPs to provide the user interface. The JSPs call EJB session beans for client session management, and the session beans call EJB entity beans to manage data in the back-end database.

The MyPortfolio files are in the directory *sample/MyPortfolio*. The file *readme.html* describes how to install and run the application in EAServer.

Using EJB CMP entity beans

EAServer includes a sample EJB 2.0 CMP entity bean, in the installation subdirectory *html/classes/Sample/cmp20sample*. The sample defines two components to demonstrate the use of container managed persistence for entity beans, including EJB-QL queries for finder and select methods. This sample also shows the use of local interfaces to call the entity bean from a session bean. For instructions on deploying and running the sample, see the *readme.txt* file included with the source files.

Using message-driven beans and JMS

This sample shows how to run message-driven beans (MDBs) and use the Java Message Service (JMS) in EAServer. JMS is a standard API for asynchronous messaging in Java. MDBs are EJB 2.0 components that respond to messages published in the message service. For information on using JMS and creating MDBs, see Chapter 31, "Using the Message Service," in the *EAServer Programmer's Guide*.

The file *html/classes/MDBSample.jar* contains all the files you need to deploy and run the sample, as well as the source code. Follow the instructions below to deploy the MDB and run the sample client. When you import the JAR file, EAServer Manager creates and configures the MDB component and also extracts the source files to the *html/classes/Sample/MDBSample* directory in your installation.

❖ **Importing the JAR file, installing the MDB, and running the JMS client**

1 If the message service is not running, configure and start it as described in Chapter 8, "Setting up the Message Service," in the *EAServer System Administration Guide*.

2 Using EAServer Manager, import the MDBSample JAR file:

   a Highlight the Packages folder.

   b Choose File | Deploy | Jaguar JAR.

   c In the Specify JAR File dialog box, click Browse and select the */html/classes/MDBSample.jar* file.

   d Click OK.

3 Install the MDB component:

   a Highlight the Jaguar | Installed Packages folder.

   b Choose File | Install Package.

c In the Package wizard, click Install an Existing Package.

d In the list of packages, highlight MDBSample, then click OK.

4 Restart the server.

5 At a command prompt, run the JMS client:

On Windows:

```
%JAGUAR%\html\classes\Sample\MDBSample\run.bat
```

On UNIX:

```
$JAGUAR/html/classes/Sample/MDBSample/run.sh
```

The console window displays the program status.

When you run the JMS client, it sends a message to the message queue *myQueue*; the MDB listens on this queue. When the MDB receives the message, it sends the message to the queue *yourQueue*, and the client receives the message from *yourQueue*. The message queues are created automatically when you run the JMS client.

Using SSL in Java clients

Clients can use SSL to enhance the security of their connection to EAServer. You can also configure servers to require that clients provide an SSL certificate when connecting to EAServer. For information on configuring EAServer to use SSL, see Chapter 12, "Security Configuration Tasks," in the *EAServer Security Administration and Programming Guide*.

EAServer provides samples to demonstrate SSL connectivity from standalone Java clients and Java applets:

- Two standalone Java client samples are provided in the *html/classes/Sample/ClientSSL* directory. The file *readme.txt* in this directory describes how to compile and run the samples.
- The applet sample is in the *html/classes/Sample/SecurityDemo* directory. The file *readme.html* in this directory describes how to compile and run the sample.

Using custom authentication

EAServer allows you to plug in your own service to implement user authentication for component and Web application access. A sample Java implementation is provided in the directory *html/classes/Sample/AuthServiceDemo*. The file *readme.html* in this directory describes how to install and run the sample authentication service.

Using the Java Authentication and Authorization API

The Java Authentication and Authorization Service (JAAS) API provides a framework and standard programming interface for authenticating users and assigning privileges. To comply with EJB 2.0 specification, EAServer supports JAAS for user authentication in standalone EJB client applications, and for authenticating resource access to J2EE connectors.

EAServer contains a sample JAAS login module in the *html/classes/Sample/JAAS* directory, with setup instructions provided in the *readme.txt* file.

For more information on JAAS, see Chapter 10, "Using the JAAS API," in the *EAServer Security Administration and Programming Guide*.

Using CORBA components and clients

This sample shows how to use the basic predefined datatypes in CORBA component calls, and includes a Java client and a Java component. The client source is in *html/classes/Sample/Intro*. Instructions for running the sample are in the file *readme.html* in this directory.

C++ samples

In addition to the C++ tutorial in this book, your EAServer installation includes the following C++ samples.

Using SSL in C++ clients

Clients can use SSL to enhance the security of their connection to EAServer. You can also configure servers to require that clients provide an SSL certificate when connecting to EAServer. For information on configuring EAServer to use SSL, see Chapter 12, "Security Configuration Tasks," in the *EAServer Security Administration and Programming Guide*.

Sample C++ clients that use SSL are provided in the *Sample/ClientSSL* directory. The file *readme.txt* describes how to build and run the clients.

Using custom authentication services

EAServer allows you to plug in your own service to implement user authentication for component and Web application access. A sample C++ implementation is provided in the *Sample/AuthServiceDemo* directory. The file *readme.txt* in this directory describes how to build, install, and run the sample authentication service.

Inspecting client SSL credentials

EAServer components can inspect the calling client's SSL credentials using the CtsSecurity::UserCredentials interface. A sample C++ component demonstrating this feature is provided in the *Sample/SecurityDemo* directory.

For information on building, installing, and running the component, see the file *readme.html* in the directory *html/classes/Sample/SecurityDemo.*

Java Pet Store

The Java Pet Store is a large, comprehensive sample application developed by Sun Microsystems to run on J2EE-compliant servers. Sun Microsystems provides Java Pet Store as part of the J2EE BluePrints series. For information on Java Pet Store and other J2EE blueprints offerings, see the J2EE Blueprints Web site at http://java.sun.com/j2ee/blueprints/.

For information on running Java Pet Store in EAServer, see the Sybase EAServer home page at http://www.sybase.com/products/easerver/.

PowerBuilder samples

While PowerBuilder is not included with EAServer, the products are integrated and work well together. You can use PowerBuilder to develop and deploy EAServer business logic components, Web applications, and EAServer clients with rich graphical user interfaces. For more information, see the *Application Techniques* manual in the PowerBuilder documentation. The following samples demonstrate EAServer application development using PowerBuilder:

- The PowerBuilder *Getting Started* manual includes a tutorial to create JSP Web-services based applicatons.
- The Sybase CodeXchange Web site includes PowerBuilder samples at http://easerver.codexchange.sybase.com/servlets/ProjectDocumentList?folderID=158

Web Services Toolkit samples

Web Service Toolkit (WST) is included with EAServer. You can use WST to develop, manage and expose Web services in EAServer. For a description of the samples and tutorials included with WST, see Chapter 10, "Using the Web Services Toolkit Samples," of the *Web Services Toolkit User's Guide*.

Samples on the Sybase Web site

You can find additional EAServer samples on the EAServer CodeXchange pages at http://easerver.codexchange.sybase.com/. The CodeXchange site allows Sybase users to share code samples and utilities for EAServer and other Sybase products.

Index

A

Adaptive Server Anywhere
  starting 51
  using in EJB tutorial 46
applets
  browser requirements for 20
  code for 22
  configuration requirements for 22
  creating 20
  debugging 28
  HTML for 26
  running 22, 28
applications
  C++ 29
  CORBA 9
  EJB 46
  Web 2
architecture
  for distributed applications 2
  of EAServer applications 1
authentication
  customization of 60, 61
  using JAAS API 60

B

BluePrints, Java 61

C

C++
  client code for 37
  compilers for 36, 37, 41
  component code 35
  custom authentication with 61
  generating files for 35
  running clients 43
  samples for 60
  tutorial 29
  using SSL in 60, 61
client
  definition of 2
clients
  C++ 29, 37
  EJB 54
  Java 15
  sample Java 57
  types of 2
code
  C++ client 37
  C++ component 35
  C++ samples 60
  client-side applet 22
  CORBA Java 60
  EJB 57
  for EJB client 54
  for EJB entity beans 48
  for EJB session beans 49
  for Java clients 15
  for JMS 58
  for MDB 58
  generating for C++ 35
  HTML 26
  Java component 14
  Java samples 57
  JavaServer Pages 57
  to load applets 26
  to use SSL in C++ 60, 61
  to use SSL in Java 59
compiling
  C++ clients 41
  C++ components 36, 37
  Java files 10
components
  C++ 29, 32
  definition of 2
  EJB 47

EJB entity beans 47
EJB session beans 49
importing classes for 48
Java 11
Java code for 14
language support for 2
properties for C++ 33
properties for Java 12
sample Java 57
session beans 49
types of 2
components, C++
building 36, 37
compiling 36, 37
components, EJB 47
components, Java 9
concepts
EAServer 1
connection caches
associating with EJB components 54
creating 53
connection screen
host name 6
jagadmin password 6
port number 6
conventions vii
CORBA
C++ tutorial for 29
creating Java applications with 9
Java samples for 60
Java tutorial for 9
creating
applets 20
connection caches 53

D

databases
for EJB tutorial 51
used with EAServer 1
debugging
applets 28
defining
C++ component 32
methods 34
development environments
Java 10, 46
distributed applications
explanation of 2
distributed architecture
explanation of 1

E

EAServer Manager
configuring 4
connecting to EAServer 5
starting 4
EJB
clients 45, 54
component properties for 48
creating components 47
entity bean 47
entity beans 45
generating files for 51
requirements for 45
sample components 45
samples for 57
session beans 45, 49
tutorial for 45
tutorial steps 46
Enterprise JavaBeans
See EJB
entity beans
code for 48
creating 47
properties for 48
examples
JMS client and MDB 58

G

generating
C++ files 35
EJB files 51
Java files 13

H

HTML
to run applets 26
HTTP
network protocol 2

I

IIOP
network protocol 2
importing
Java classes 48
interfaces, user 1

J

J2EE
samples for 61
JAAS
sample for 60
jagadmin
connecting to EAServer Manager 6
Jaguar Manager
See EAServer Manager
Java
browser plug-ins for 20
compiling 10
CORBA samples for 60
custom authentication with 60
defining components 11
development environments 10, 46
generating files for 13
importing classes for 48
samples for 57
skeletons 13
stubs 13
tutorial for 9
using SSL in 59
Java Pet Store
J2EE sample application 61
Java Plug-in
using 20
JavaServer Pages
samples 57
JDK
for Web browser 20
JMS
definition of 58
sample client and MDB 58
samples using 58

L

listener
definition of 3

M

MDB
definition of 58
sample 58
samples using 58
methods
defining 13, 34
multitier
application development overview 1
MyPortfolio
sample application 57

O

overview
of multitier application development 1

P

packages
definition of 2
parameters, method
defining 13, 34
Pet Store
J2EE sample application 61
properties
for C++ components 33
for connection caches 53
for EJB components 54

for entity bean components 48
for Java components 12
for session bean components 50
protocols
HTTP 2
IIOP 2
listeners for 3
proxy objects
definition of 2

R

refreshing servers 6
requirements
for EJB tutorial 45
for Java 10
resource references
for EJB components 54
restarting servers 6
running servers 3

S

samples
besides tutorials 57
C++ 60
CORBA Java 60
EJB 57
for custom authentication 60, 61
for JMS 58
in the installation 57
J2EE 61
JAAS API 60
Java 57
JavaServer Pages 57
MDB 58
MyPortfolio 57
on the Sybase Web site 62
using SSL 59, 60, 61
security
JAAS sample for 60
servers
refreshing 6
restarting 6
running 3
shutting down 6
starting 3
session beans
code for 49
component properties for 50
creating 49
skeletons
C++ 35
EJB 51
explanation of 2
generating Java 13
SSL
C++ samples for 60, 61
Java samples for 59
starting servers 3
stubs
C++ 35
definition of 2
EJB 51
generating Java 13
Sybase Central
management with 5

T

tutorial, C++
client code for 37
defining the component 32
generating files for 35
running 43
server-side code 35
tutorial, EJB
creating the client 54
creating the entity bean 47
creating the session bean 49
generating files for 51
steps for 46
tutorial, Java
client program for 15
component code for 14
defining methods for 13
defining the component 11
generate files for 13
tutorials

C++ 29
EJB 45
Java 9, 45
typographical conventions vii

U

user interfaces, types of 1

W

Web
finding samples on 62
running applications on 2
Web application, definition of 2
Web browser, running applets in 20