Contents

About This Book ................................................................................................................ .......... vii

CHAPTER 1 Introduction to Jaguar CTS............................................................ 1
   Terminology and concepts ............................................................... 2
   Terminology............................................................................... 2
   Concepts ................................................................................... 3
   Developing a Jaguar application ............................................... 4
   Jaguar runtime environment...................................................... 4
   Starting Jaguar ................................................................................. 6
   Using the JagRepair server ....................................................... 9
   Setting the BOOTCLASSPATH and BOOTLIBRARYPATH variables ........................................................................... 10
   Setting the JAGUAR_RANDOMSEED variable ...................... 10
   Using Jaguar Manager and Security Manager............................... 11
   Starting Jaguar Manager and Security Manager..................... 12
   Profile Manager ....................................................................... 13
   Disconnecting from Jaguar ...................................................... 14
   Shutting down Jaguar .............................................................. 14
   Verifying your environment ............................................................ 14
   Using the sample Jaguar application ............................................. 16

CHAPTER 2 Tutorial: Java Components and Java Clients ............................. 17
   Overview of the sample application................................................ 18
   Tutorial requirements ..................................................................... 19
   Java compiler scripts...................................................................... 20
   Creating the application ................................................................. 21
   Start the Jaguar server and Jaguar Manager.............................. 21
   Define a package, component, and method.............................. 21
   Generate stubs using the Jaguar Manager..................................... 23
   Compile the stub code............................................................. 24
   Write the server-side code ........................................................ 24
   Write the client-side code ........................................................ 25
   Create the HTML page and run the sample ......................... 29
About This Book

Subject
This book contains information about running Jaguar servers and connecting to Jaguar servers from Jaguar Manager and Security Manager. It also provides tutorials that help you create basic Jaguar applications.

Audience
This book is for anyone responsible for configuring the Jaguar runtime environment, establishing security, or for creating and deploying packages and components on a Jaguar server.

How to use this book
Chapter 1, “Introduction to Jaguar CTS” introduces you to the terminology and concepts associated with Jaguar CTS. It also includes:

• Instructions for starting a Jaguar CTS server.
• Instructions for starting Sybase Central and connecting to the Jaguar plug-ins, Jaguar Manager, and Security Manager. Use Jaguar Manager to configure the Jaguar server and manage packages and components. Use Security Manager to manage the client and server certificates that are required for Secure Sockets Layer (SSL) connections.
• Information on environment variables and settings required by the Jaguar server.
• Instructions for running the Sybase Virtual University (SVU) sample application.

Chapter 2, “Tutorial: Java Components and Java Clients” describes how to create a simple Java component and a Java applet client that calls the component methods.

Chapter 3, “Tutorial: C++ Components and C++ Clients” describes how to create a simple C++ component and a C++ client that calls the component methods.

Chapter 4, “Tutorial: ActiveX Components” describes how to create a simple ActiveX component and test it by running a Java applet client that invokes a component method.

Chapter 5, “Tutorial: Security Configuration” describes how to configure Jaguar and Netscape to use SSL features.
Note
The tutorials in this book are intended for first-time users. After completing these tutorials, you should be familiar enough with Security Manager and Jaguar Manager to be able to write simple Jaguar CTS applications.

Related documents
The Enterprise Application Server Installation Guide contains installation instructions. The installation CD includes a README.txt file that contains the latest information about Jaguar.

What's New in Jaguar CTS contains information about new features in Jaguar.

The Jaguar CTS System Administration Guide contains information about configuring name services, security, clusters, load balancing, and failover for Jaguar servers.

The Jaguar CTS Programmer's Guide contains detailed information about C/C++, Java, and ActiveX components, Java stub classes and stub generation, session management, event handlers, and connection management, as well as Jaguar's programming models. It provides information on Jaguar-supported datatypes, result-set handling, and compiling dynamic link libraries (DLLs).

The Jaguar CTS API Reference contains reference pages for the Java classes and interfaces and C routines used by Jaguar. It is available in HTML format in the %JAGUAR%/html/docs directory of your Jaguar installation.

The Sybase Web page provides white papers and other product information. See:

• The EA Server Web site at http://www.sybase.com/products/easerver/
• The EA Studio Web site at http://www.sybase.com/products/eastudio/

Other sources of information
Use the Sybase Technical Library CD and the Technical Library Web site to learn more about your product:

• Technical Library CD contains product manuals and technical documents and is included with your software. The DynaText browser (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.
About this book

Refer to the Technical Library Installation Guide in your documentation package for instructions on installing and starting Technical Library.

- Technical Library Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser.

  To use the Technical Library Web site, go to www.sybase.com and choose Documentation, choose Technical Library, then choose Product 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

Introduction to Jaguar CTS

About this chapter

This chapter explains some of the features of the Jaguar Component Transaction Server™ (CTS).

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminology and concepts</td>
<td>2</td>
</tr>
<tr>
<td>Starting Jaguar</td>
<td>6</td>
</tr>
<tr>
<td>Using Jaguar Manager and Security Manager</td>
<td>11</td>
</tr>
<tr>
<td>Shutting down Jaguar</td>
<td>14</td>
</tr>
<tr>
<td>Verifying your environment</td>
<td>14</td>
</tr>
<tr>
<td>Using the sample Jaguar application</td>
<td>16</td>
</tr>
</tbody>
</table>
Terminology and concepts

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 Jaguar Manager. For detailed information on Jaguar application development, see the Jaguar CTS Programmer’s Guide.

Terminology

A Jaguar application consists of one or more packages and a client application or applet. Packages consist of components, and components are made up of one or more methods.

- Jaguar can host, manage, and execute components such as ActiveX programmable objects, JavaBeans, or CORBA-compliant components. In the Jaguar environment, a component is simply an application object that consists of one or more methods. Jaguar components typically execute business logic, access data sources, and return results to the client. Clients (applets) create an instance of a component and execute methods associated with that component. Components run only within a Jaguar server.

- 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, grouping together application resources for ease of deployment and management.

  Jaguar supports the following types of components:

  - ActiveX
  - C
  - CORBA C++
  - Java

- A stub is a Java class or a C++ stub generated by Jaguar Manager and 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** acts as the interface between the Jaguar runtime environment and the user code that implements the method. Skeletons are compiled and linked with each of the components, and at runtime they enable Jaguar to locate and invoke an appropriate method.

Jaguar transparently maintains a **session** between a client application and the Jaguar server. Unlike a typical HTTP scenario, where a new connection is created for each request and response, sessions allow a browser to maintain a connection with the server across a multiple request-response cycle.

### Concepts

You can develop and distribute a Jaguar application across the network. Jaguar CTS implements a **three-tier or multitier** distributed computing architecture. In this model, three distinct elements work together to give users access to data:

- Client-side applet or application
- Middle-tier components
- The back-end database

Java applets are downloaded to clients, which instantiate components on the server. Client applications are installed on client machines, from which they also instantiate components on the server.

An applet or application manages presentation and interaction with an end user. Middle-tier components, which run in Jaguar CTS, handle much of the application processing. Finally, the database stores, manages, and processes data.

If the client is an applet, users find and launch applications from traditional HTML pages. Instead of simply loading a static page, Jaguar downloads an executable applet to the individual’s browser. If the client is an already-installed application, the user launches the application from his or her machine. Clients communicate directly with an application component running in the middle tier. Server components access data from one or more databases, apply business logic, and return results to the client applet for display.
When a proxy object is created on the client applet, it instantiates a corresponding component registered with the Jaguar server. On the server side, a component is instantiated in response to a request from the proxy object running in the client environment. A method on a component is executed when it is invoked by a proxy object on the client applet.

Developing a Jaguar application

There are three basic steps involved in creating and deploying a Jaguar application that employs a Java applet as a client. For information on other types of Jaguar clients, see the *Jaguar CTS Programmer’s Guide*.

❖ To create and deploy a Jaguar application:

1. Define packages, components, and methods. The Jaguar Manager is Jaguar’s GUI interface that allows you to easily define the packages, components, and methods that Jaguar clients use to run an application. Jaguar Manager generates:
   - Client-side stub files – Stubs contain interface information used by the client to invoke Jaguar component methods.
   - Server-side skeleton files – Skeletons provide the interface information of each component method.

2. Once you have generated the stubs and skeletons, write the Java classes that, once linked with the stub files, form the basis of your downloadable applet.

   Develop the server-side components that link with the skeletons to form the business logic of your servlet. Jaguar supports many of the integrated development environment (IDE) tools available today.

3. Deploy the application. You can register components on any Jaguar server. Because Jaguar CTS is also a Web server, you can write an HTML page for your applet and install it on Jaguar.

Jaguar runtime environment

A typical Jaguar application has an applet or HTML page associated with it. Once you build and deploy such an application, it runs in the following fashion:
1 Jaguar receives an HTTP request and downloads the requested HTML page or applet. Included with the applet are the Java stubs, which through a proxy, instantiate components and invoke the methods on those components.

2 The client establishes a session with Jaguar. The session, unlike an HTTP connection, allows the client and Jaguar to maintain a connection throughout the transaction.

3 The client creates a component instance through a client-side proxy. The proxy used depends on the type of component being instantiated. Jaguar validates the user against the component’s access list. If the user is validated, the dispatcher checks the location and status of the component and creates an instance.

4 The client invokes the component’s business logic by executing its methods.

5 The component may interact with remote databases. If it does:
   • The component obtains a connection to the database using Jaguar’s connection caching feature.
   • Jaguar checks the component’s transaction property. If the component is marked as transactional, Jaguar ensures that remote-database commands execute as part of a larger transaction.

6 Jaguar returns the results from the database to the client.

7 The client indicates that it has completed the operation. Jaguar destroys the component instance or returns it to a pool for future client instantiations. The client disconnects from Jaguar.
Starting Jaguar

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

If the preconfigured server is not installed as an NT service, you can start it using the Windows Start menu, under Start | Programs | Sybase | Jaguar CTS. Choose one of the options below:

Jaguar Server –

• Starts the normal-mode server using JDK 1.1.
• Jaguar Server (JDK 1.2) – Starts the normal-mode server using JDK 1.2.
• Jaguar Server (debug) – Starts the debug-mode server using JDK 1.1.
• Jaguar Server (debug JDK 1.2) – Starts the debug-mode server using JDK 1.2.

Debug and JDK 1.2 servers
Debug-mode servers allow you to remotely debug components from tools that support Jaguar component debugging, such as PowerBuilder or PowerJ. You cannot run the debug-mode server unless you installed the debug libraries and binaries. The debug server cannot run as an NT service.

Jaguar can run with your choice of Java runtime engines, including Sun JDK 1.1 or 1.2, or the Microsoft Java Virtual Machine. You must run Jaguar with JDK 1.2 to use the full set of available features. Web applications, Java Server Pages, and EJB 1.1 components require JDK 1.2.

To start a user-defined server, you must first create a server. See “Creating or deleting a server” in the Jaguar CTS System Administration Guide for instructions.

Then change to the %JAGUAR% directory, and run one of the following commands, where Server_name is the name of the Jaguar server you want to start:

• To start the server in normal mode using JDK 1.1:
  bin\serverstart Server_name
• To start the server in normal mode using JDK 1.2:
  devbin\serverstart_jdk12 Server_name
• To start the server in debug mode using JDK 1.1:
Starting servers that are installed as NT services

If a Jaguar server is installed as an NT service, the server is started automatically whenever you start NT. To start a service manually, you can use the Services dialog in the Windows NT Control Panel, as follows:

1. On the NT desktop, select Start | Settings | Control Panel.
2. Double-click the Services icon.
3. In the list of Services, find the name of your Jaguar server (for example, the preconfigured server is “Jaguar”). To stop the server, highlight the server name and click Stop. To start the server, highlight the name and click Start.

You can also run this command in the Jaguar bin subdirectory:

```
serverstart servicename -start
```

where `servicename` is the name of the server. To stop the server, use:

```
serverstart servicename -stop
```

Installing servers as NT services

You should install production servers as NT services, so the Jaguar server starts automatically when NT starts. Do not install development servers as NT services, since you will often want to run the debug server on your development machine, and the debug server cannot be run as an NT service.

❖ To install a server as an NT service:

1. If the server is not running, start it on the desktop.
2. Start Jaguar Manager and connect to the server using Jaguar Manager. See “Starting Jaguar Manager and Security Manager” on page 12 for instructions.
3. In Jaguar Manager, display the Server Properties dialog box by highlighting the server’s icon and choosing File | Server Properties. On the Java VM tab, select the Java runtime to be used by the server.

Some Jaguar features require JDK 1.2

You must run Jaguar with JDK 1.2 to use the full set of available features. Web applications, Java Server Pages, and EJB 1.1 components require JDK 1.2.
4 Shut down the Jaguar server by highlighting its icon and choosing File | Shutdown.

5 Depending on which JDK you configured, run one of the following commands in the Jaguar bin subdirectory, specifying the server name in place of server:

For JDK 1.2:

```
serverstart_jdk12 server -install
```

For JDK 1.1:

```
serverstart server -install
```

For the Microsoft Java Virtual Machine:

```
serverstart_msvm server -install
```

❖ To remove a server from the list of NT services:

- Run the command:

```
serverstart servicename -remove
```

where servicename is the name of the server as displayed in the Services dialog in the Windows NT Control Panel.

Complete syntax for serverstart

This section lists the complete syntax of the serverstartXXX batch files, located in the Jaguar bin subdirectory.

```
serverstartXXX [ servername [ options ]]
```

where

- XXX specifies the JDK.
- servername specifies the name of the Jaguar server to start, as displayed in Jaguar Manager. You must supply a server name if you supply options. If you do not specify any arguments, the default behavior matches the following command, which starts the preconfigured Jaguar server in a console window:

```
serverstart Jaguar -c
```

- option is one of:
• -c (the default) starts the server as a console application. You must use this option if you have not installed the server as an NT service.
• -install installs the specified server as an NT Service.
• -remove uninstalls the specified server from the list of NT Services.
• -removeandinstall reinstall the specified server into the list of NT Services.
• -start starts a server that has been installed as an NT service. When installed as an NT service, Jaguar starts automatically when NT starts. Use this option only after you have stopped or shut down the server.
• -stop shuts down a server that has been installed as an NT service. Use this option to restart the server. You can also stop the server using the NT Services dialog in the Control Panel, or from Jaguar Manager.
• -v prints server version information to the console, then exits.

Using the JagRepair server

You can use the JagRepair server when you are unable to start your Jaguar server; for example, if you have specified incorrect Object Transaction Service (OTS) settings which prevents the server from starting and makes it impossible to correct the problem. The JagRepair server is read-only and provided for repair purposes only.

❖ To start and connect to the JagRepair server:
1 At the command line, enter:
   %JAGUAR%\bin\serverstart JagRepair

2 Start Jaguar Manager.

3 Select Tools | Connect | Jaguar Manager and enter:
   • jagadmin as the User Name
   • localhost as the Host Name
   • 9000 as the Port Number

4 Click Connect.
Setting the **BOOTCLASSPATH** and **BOOTLIBRARYPATH** variables

When you are running the JDK 1.2 Jaguar server, you may need to edit the **BOOTLIBRARYPATH** and **BOOTCLASSPATH** environment variables.

Set **BOOTLIBRARYPATH** if you use Java classes that call native code in DLLs or shared libraries. Add the locations of these DLLs or libraries to the **BOOTLIBRARYPATH** environment variable. The syntax for setting this environment variable is the same as for setting the **PATH** variable.

Set **BOOTCLASSPATH** if your Jaguar components require Java classes that are not in the standard Jaguar locations (the Jaguar *html/classes* and *java/classes* subdirectories). For Java components and Web applications, you can also specify classes to be loaded on a per-component or per-Web-application basis. See the *Jaguar CTS Programmer’s Guide* for more information.

Setting the **JAGUAR_RANDOMSEED** variable

Jaguar requires a random seed to initialize the random number generation used in cryptographic algorithms. The data used as the seed for the random number generation depends on your platform. On NT, Jaguar accesses the contents of the HKEY_PERFORMANCE_DATA Registry entry. On UNIX machines, Jaguar accesses the process, virtual memory, and network statistics. However, you can set the **JAGUAR_RANDOMSEED** variable to improve Jaguar server performance without diminishing the randomness of the seeding data.

**JAGUAR_RANDOMSEED** determines the algorithm as follows:

- If you set the variable to the name of an accessible file, Jaguar reads this file to obtain random seeding data. Use this technique to avoid using system performance data as the seed. You must specify the name of a file that contains frequently changing contents, such as the access log of a busy server.

- If you set the variable to a value that does not match the name of an accessible file, Jaguar obtains seeding data by calling a sequence of system routines to obtain performance data.

❖ To set **JAGUAR_RANDOMSEED**:

1. Identify a file that contains suitably random data, such as the access log of a busy server.
The file can contain text or binary data, but the contents should change randomly and often. The file must be accessible from the machine and account that are used to run the Jaguar server.

2 Create a new system environment variable called $JAGUAR_RANDOMSEED$ and set its value to the full path of this file.

3 Restart the Jaguar server.

Using Jaguar Manager and Security Manager

Jaguar Manager and Security Manager run within Sybase Central. Use Jaguar Manager to configure the Jaguar server and to define and deploy software components and packages. Use Security Manager to manage SSL certificates, the test CA, and public and private key pairs for the Jaguar server.

Note You can connect to Jaguar server version 3.6 from Jaguar Manager and Security Manager version 3.6, but you cannot connect to earlier versions of the Jaguar server from Jaguar Manager or Security Manager version 3.6.

To use Jaguar Manager or Security Manager, you must be the jagadmin user or belong to the Admin role. If you are connecting for the first time, use jagadmin as the user name and leave the password blank.

For information on the following topics, see the Jaguar CTS System Administration Guide:

- For additional information on Jaguar administrative privileges, see “Admin role in Jaguar.”
- For additional security you can establish a password for the jagadmin user. See “Administration password and OS authentication.”
- Jaguar comes preconfigured to allow you to run the sample applications and to create the applications used in the tutorials. You may need to configure Jaguar further to run other applications.
- Detailed information about Security Manager.

A Jaguar server must be running before Sybase Central can connect to it.
Starting Jaguar Manager and Security Manager

Jaguar Manager and Security Manager are plug-ins for Sybase Central. You must start Sybase Central first, then start the plug-ins from within it.

❖ To start Sybase Central:
   • Double-click the Jaguar Manager icon in the Jaguar CTS program window, or select Start | Programs | Sybase | Jaguar CTS | Jaguar Manager.

Once the Sybase Central screen appears you can start Jaguar Manager, Security Manager, or both.

❖ To start both Jaguar Manager and Security Manager:
   2. On the Login screen:
      a. Enter the jagadmin user name.
      b. Enter the jagadmin password if you have established one. Otherwise, leave this field blank.
      c. Specify the host name or IP address. The default is localhost. You should change the default setting to the actual machine name or IP address of your machine. See “Preconfigured listeners” in the Jaguar CTS System Administration Guide for more information.
      d. Verify the Jaguar port number.
         The machine name or IP address and port number correspond to the Jaguar host entry and IIOP port number defined in the listener.
      e. Enter the PIN to connect to the Sybase PKCS #11 token. The default PIN is sybase. See “Changing the user PIN” in the Jaguar CTS System Administration Guide for more information.
      f. Click Connect.
   3. Click the Sybase Central icon. You now have access to both Security Manager and Jaguar Manager and the resources that they manage.

❖ To start Jaguar Manager only:
   1. Select Tools | Connect | Jaguar Manager.
   2. On the Login screen, enter the jagadmin user name. Specify the host name or IP address, verify the Jaguar port number, and click Connect. The machine name or IP address and port number correspond to the Jaguar host entry and IIOP port number defined in the listener.
If Jaguar Manager is running and you want to start Security Manager:
2. Enter the PIN to connect to Sybase PKCS #11. The default PIN is `sybase`.

Standalone versions of Jaguar Manager and Security Manager

If you install the Jaguar CTS client software you can run standalone versions of Jaguar Manager and Security Manager. This is helpful for managing a server, generating stubs and skeletons, managing certificates, and so on, from a Jaguar client.

To start the standalone Jaguar Manager, go to the Jaguar client installation directory and enter:

```
bin\sajagmanager.bat
```

Profile Manager

You can also use the Profiles field to log in to Jaguar Manager. The Profiles field allows you to select predefined login profiles to speed up the connection process. Define profiles using the Profile Manager, then select the profile from the drop-down list in the login window and enter a password to connect to the port defined in the profile.

To define a profile from the login window:
1. Click the Profile Manager button.
2. Click Add and define the following fields:
   - **Profile Name** – the name of the profile that displays in the Profiles drop-down list.
   - **User Name** – the name of the user connecting to the port; for example, jagadmin.
     
     To log in using a profile, you must provide a password in the login window that corresponds to the user name of the login profile.
   - **Host Name** – the host name to which you are connecting.
   - **Port Number** – the port on the host to which you are connecting.
3. Click OK to add the profile.

To delete a profile, highlight the profile and click Delete.
Shutting down Jaguar

Disconnecting from Jaguar

Sybase Central allows you to disconnect Jaguar Manager from a Jaguar server so that you can connect to another server, or reconnect to the same server, without restarting Jaguar Manager.

❖ To disconnect from Jaguar:
  • Select Tools | Disconnect | Jaguar Manager.

Shutting down Jaguar

❖ To shut down a Jaguar server from Sybase Central:
  1 Select Servers | server_name, where server_name is the Jaguar server to which Jaguar Manager is connected. You must be connected to a server in order to shut it down.
  2 Select File | Shutdown server_name.
    Unless you have changed the listener address for the server, you can remain logged in to Jaguar Manager and/or Security Manager and resume work after you have restarted the server.
  3 If you have changed the network listener settings (host name and port) for that server, you must exit Sybase Central and reconnect using the new host name and port number.

Note
You can shut down only the server to which Sybase Central is connected.

Verifying your environment

Your environment is set up automatically by the installation program. However, your environment may have been changed, for example, if you installed additional software. Use the information here to troubleshoot your installation.

If you have any problems running the Jaguar server, sample programs, or Sybase Central:
• Verify that the Jaguar installation was performed as user “Administrator.”

• Check the Jaguar server log file (srv.log in the bin subdirectory) for error messages.

• Verify the Jaguar HTTP, IIOP, and TDS port settings. Default listeners are defined for each protocol, but you may have changed them. See the Jaguar CTS System Administration Guide for more information.
  • TDS is the port used by Jaguar to accept TDS requests. The default is 7878.
  • HTTP is the HTTP port used by Jaguar to listen for HTTP requests. The default is 8080.
  • IIOP is the port used by Jaguar to accept IIOP requests. The default is 9000.

Note  The default entries use localhost as the host name. You must edit the default host name settings if you want clients on other machines to access your Jaguar server. If you configure invalid listener addresses (for example, by specifying an invalid host name), Jaguar attempts to use an alternate address. When this occurs, the new address is recorded in the server log file.

• Verify the LIB environment variable.
  If you use the Microsoft Visual C++ compiler, the following string should be listed in your LIB variable for compiling your C/C++ application (assuming the C compiler has been installed in C:\MSDEV):

  c:\Powersoft\jaguar\lib;c:\MSDEV\lib

• Verify the INCLUDE environment variable.
  If you use the Microsoft Visual C++ compiler, the following string should be listed in your INCLUDE variable for compiling your C/C++ application (assuming the C compiler has been installed in C:\MSDEV):

  c:\Powersoft\jaguar\include;c:\MSDEV\include
Using the sample Jaguar application

In addition to the tutorials described in this book, Jaguar comes with a Sybase Virtual University (SVU) sample application. The application simulates an online registration and class enrollment application for college students in their junior and senior years. It allows students to enroll in classes, obtain loans based on financial need, and buy books based on classes they have enrolled in. The application consists of one applet and three components.

You must install and start the Adaptive Server Anywhere (ASA) database to run the sample. See the Enterprise Application Server Installation Guide for more information.

As an end user, you will run the sample application by running a client applet that connects to the Jaguar server, and instantiates several Jaguar components. The necessary applet and components are automatically installed with Jaguar.

With Jaguar and ASA running, download the Sybase Virtual University applet by clicking the icon for Sybase Virtual University located on the Jaguar CTS index page at http://hostname:8080. The default hostname is localhost. Log in to the application by acquiring a student ID. The components update data on the ASA database. Source code for the Java applets and Java components are located in the sample directory.
**Tutorial: Java Components and Java Clients**

**About this chapter**

In this tutorial, you will create a Java component, install it in the Jaguar server, and create a Java applet that connects to Jaguar and calls a method in the component.

**Contents**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of the sample application</td>
<td>18</td>
</tr>
<tr>
<td>Tutorial requirements</td>
<td>19</td>
</tr>
<tr>
<td>Java compiler scripts</td>
<td>20</td>
</tr>
<tr>
<td>Creating the application</td>
<td>21</td>
</tr>
</tbody>
</table>

**For more information**

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

- Chapter 6, “Creating Java Components”
- Chapter 11, “Creating CORBA Java Clients”
Overview of the sample application

The application performs the following steps:

1. The client-side applet, developed with Java, instantiates the middle-tier Java component, JavaArithmetic.
2. The applet calls the multiply method in JavaArithmetic.
3. The multiply method computes the product of the input values, then returns the result.
4. The applet displays the result for the end user.
Tutorial requirements

To create the tutorial application, you need:

- The Jaguar CTS software

  If you have not already installed Jaguar, follow the instructions in the *Enterprise Application Server Installation Guide* to install the software.

- Java development environment

  Jaguar comes with two versions of Sun Microsystems Java Developer’s Kit: JDK 1.0.2 and JDK 1.1\(x\), where \(x\) is the latest JDK release that has been tested with Jaguar. “Java compiler scripts” on page 20 describes script files that will set the environment to run the JDK compilers that are included in your Jaguar installation.

  In the sample application, you will be developing a Java applet that will handle user interaction. For the applet development, you may use any of the off-the-shelf Java rapid application development (RAD) tools including:

  - Sybase® Power J™
  - Symantec Visual Cafe
  - Microsoft Visual J++

To run the tutorial applet, you must have:

- A Web browser that is compatible with JDK 1.1

  Netscape 4.\(x\), Internet Explorer 4.0, or Sun’s HotJava browser have all been tested with this tutorial.

**Note**

Older versions of the Netscape and Microsoft browsers support only JDK 1.0.2. You may need to create clients that are compatible with JDK 1.0.2 to satisfy your site’s deployment requirements. See the *Jaguar CTS Programmer’s Guide* for more information.
Java compiler scripts

This tutorial assumes that you are using the Java compiler that is provided by your Jaguar installation; however, you can use any compiler as long as it produces bytecode that is compatible with JDK 1.1.

Jaguar provides a compilation batch file, j11.bat, in the bin subdirectory. j11.bat compiles Java source files with the JDK 1.1 command line compiler. The tutorial steps use this batch file, but you may also compile source with an IDE that supports JDK 1.1, such as PowerJ.
Creating the application

To create and run the sample application:

1. Start the Jaguar server and Jaguar Manager
2. Define a package, component, and method
3. Generate stubs using the Jaguar Manager
4. Compile the stub code
5. Write the server-side code
6. Write the client-side code
7. Create the HTML page and run the sample

Start the Jaguar server and Jaguar Manager

❖ Start the Jaguar server:
• If the Jaguar server is not already running, follow the instructions under “Starting Jaguar” on page 6 to start the server.

❖ Start Jaguar Manager:
• If Jaguar Manager is not already running, start it as described in “Starting Jaguar Manager and Security Manager” on page 12.

Define a package, component, and method

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

For complete information on creating packages, components, and methods, see the Jaguar CTS Programmer’s Guide.

Define a new package

In Jaguar, a package is a unit of deployment for a group of components that perform related tasks. A component must be installed in a package before it can be instantiated by applications. All components created in the Jaguar tutorials are installed in the Tutorial package.
❖ **Create the Tutorial package if it does not exist:**

1. Navigate to:
   
   Jaguar Manager | Servers | Jaguar

2. In the right panel, click Installed Packages. If the Tutorial package is displayed, skip to the next section, “Define and install a new component” on page 22.

3. 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.
   
   When you click on the Installed Packages expansion sign, it now includes Tutorial.

Define and install a new component

❖ **Using Jaguar Manager, define a new component for the package you just created:**

1. Click on the Tutorial package.

2. Select File | Install Component.

3. In the Component wizard, select Create and Install a New Component.
   
   For the component name, enter JavaArithmetic.

4. Click Finish.
   
   You see the Component Properties window.

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Tutorial Java component</td>
</tr>
<tr>
<td>Component Type</td>
<td>Java - CORBA</td>
</tr>
<tr>
<td>Java Class</td>
<td>Sample.Intro.JavaArithmetic.JavaArithmeticImpl</td>
</tr>
</tbody>
</table>

6. Leave the remaining fields at their default settings.

7. Click OK.
Define the multiply method

❖ After the component is defined, define methods for it that the client will call:

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.
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 using the Jaguar Manager

Once you have created the package, component, and methods, you need to generate the stub files for the component. The client-side applet uses the stubs to invoke the server-side component methods.

❖ Generate the stub files for the component:
1. Click on 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:
• Select CORBA 2.2 from the drop-down list.
• Select JDK 1.1 for Java version.
• In the Java Code Base field, type the full path to your Jaguar html\classes subdirectory, for example:
  
  %JAGUAR%\html\classes

4 Unselect Generate C++ Stubs.

5 Select Generate Skeletons.

Under Skeletons in the Java Code Base field, type the full path to your Jaguar java\classes subdirectory, for example:

  %JAGUAR%\java\classes

6 Click Generate.

Compile the stub code

Change to the Jaguar html\classes\Tutorial subdirectory and compile the generated files using a JDK 1.1 compiler, for example:

    cd %JAGUAR%\html\classes\Tutorial
    d:\j11 *.java

Write the server-side code

At this point, Jaguar Manager has created server-side implementation files in the following directory:

    %JAGUAR%\java\classes\Sample\Intro\JavaArithmetic

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

❖ Complete the component implementation and compile the classes:

1 Rename JavaArithmeticImpl.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:

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

2. Save your changes.

3. Compile the component skeleton and implementation files using a JDK 1.1 compiler—for example:
   
   cd %JAGUAR%/html/classes/Sample/IntroJavaArithmetic
   d:\j11 *.java

Write the client-side code

In the html/classes subdirectory of your Jaguar 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 subdirectory of your Jaguar installation. Here is the source for JAClient.java:

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

import org.omg.CORBA.*;
import org.omg.CosNaming.*;  // CORBA CosNaming interfaces
import org.omg.CosNaming.NamingContextPackage.*;  // CosNaming user exceptions
import SessionManager.*;
import java.awt.*;
import Tutorial.*;  // Package for component 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’s name (relative to the server’s
// initial name context), of the form package/compon
ent
String _compName = "Tutorial/JavaArithmetic";

// 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);

    try {

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

        // Initialize the ORB. Note that the org.omg.COR
        BA.ORBClass
        // property must be set in applet parameters.
//
java.util.Properties props = new java.util.Properties();
ORB orb = ORB.init(this, props);

NamingContext nc = null;
org.omg.CORBA.Object objRef = null;
objRef = orb.resolve_initial_references("NameService");
nc = NamingContextHelper.narrow(objRef);

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

//
// Create a stub object instance for the
// Tutorial/JavaArithmetic Jaguar component.
//
NameComponent compNc[] =
    { new NameComponent(_compName,"") };
Factory compFactory =
    FactoryHelper.narrow ( nc.resolve(compNc) )
;
    _comp = Tutorial.JavaArithmeticHelper.narrow(
        compFactory.create("Guest", "GuestPassword")
    );

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

} catch (NotFound nfe) {
    // This can happen if you have installed
    // the tutorial component under a different
    // package or component name in Jaguar Manager
    System.out.println("Error: Component " + _compName
        + " not found. Check the package and compone
        nt " + "name in Jaguar Manager.\n");
    System.out.println("NotFound exception details :
")
    nfe.printStackTrace();
    _comp = null;
} catch (org.omg.CORBA.UserException ue) {
// Check for other CosNaming exceptions
System.out.println("CORBA CosNaming exception: \
" + ue.toString());
ue.printStackTrace();
_comp = null;
} catch (org.omg.CORBA.SystemException se) {
    System.out.println(
        "Received CORBA system exception " \
        +"while instantiating component:\n" \
        + se.toString() );
    se.printStackTrace();
    _comp = null;
}
} // 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("\n");
        return;
    }

    // Call the server component method
    double ml = 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 this application using a JDK 1.1 compiler:

j11 JAClient.java

Create the HTML page and run the sample

In a text editor, create an HTML file that calls the applet. Name the file runjavatut.html and create it in the html/classes/TutorialApps subdirectory in your Jaguar installation.

A copy of this file is provided in your Jaguar installation directory, in the html/docs/tutorial subdirectory. You can copy this file to your html/classes/TutorialApps as an alternative to creating one. Here is the text of runjavatut.html:

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

<applet
    codebase="/classes"
    code="TutorialApps/JAClient.class"
    width=600 height=400>
    <h2>This would be a Cool Applet, but you are not running a Java enabled browser...
</h2>
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 Jaguar, edit the port number in the HTML file to match.

Run the sample

If you have not restarted the Jaguar server since creating the JavaArithmetic component, do so now before running the sample.

Configuring browsers to run applets on development machines

If you are running the sample client on a machine where EAServer has been installed, you must configure your browser to eliminate conflicts between Jaguar classes that are downloaded with the applet and classes that are loaded from the local classpath. The simplest way to do this is to run the browser from a batch file as described here:

1. Locate the browser executable, and create a batch file in the same directory with the same base filename, 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" %*
```
3 Change shortcuts that run the browser so that they launch the batch file
instead of the browser executable.

❖ **To run the sample:**

1 Start a Netscape 4.x, Internet Explorer 4.x, or HotJava Web browser. If
necessary, you can run the applet on a different machine than the Jaguar
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 Jaguar server’s host
name for `host`:

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

As installed, Jaguar 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 Jaguar method. The return value (product) 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 Jaguar server log file (`srv.log` in the Jaguar `bin` subdirectory).

If the applet connects to the server and instantiates the component successfully,
you will see ... created Jaguar session and ... created
component instance in the browser’s Java console.
CHAPTER 3

Tutorial: C++ Components and C++ Clients

About this chapter

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

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of the sample application</td>
<td>34</td>
</tr>
<tr>
<td>Tutorial requirements</td>
<td>35</td>
</tr>
<tr>
<td>Creating the application</td>
<td>36</td>
</tr>
</tbody>
</table>

For more information

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

- Chapter 7, “Creating CORBA C++ Components”
- Chapter 13, “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 Jaguar CTS software
  For installation instructions, see the Enterprise Application Server Installation Guide.
- A C++ development environment, such as:
  - Microsoft Visual C++
  - Power++™
Creating the application

To create and run the sample application:

1. Start the Jaguar server and Jaguar Manager
2. Define a package, component, and method
3. Generate stubs, skeletons, and implementation templates
4. Write the server-side code
5. Write the client-side code
6. Compile the client executable
7. Run the client executable

Start the Jaguar server and Jaguar Manager

❖ Start the Jaguar server:
   • If the Jaguar server is not already running, follow the instructions under “Starting Jaguar” on page 6 to start the server.

❖ Start Jaguar Manager:
   • If Jaguar Manager is not already running, start it as described in “Starting Jaguar Manager and Security Manager” on page 12.

Define a package, component, and method

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

For complete information on configuring packages, components, and methods, see the Jaguar CTS Programmer’s Guide.

Define a new package

In Jaguar, a package is a unit of deployment for a group of components that perform related tasks. A component must be installed in a package before it can be instantiated by applications.
Chapter 3  Tutorial: C++ Components and C++ Clients

❖ Create the *Tutorial* package if it does not exist:
1  Navigate to:

   Jaguar Manager | Servers | Jaguar

In the right panel, click Packages. If the Tutorial package is displayed, skip to the next section, “Define and install a new component” on page 37.

2  Select File | Install Package.

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

   For the package name, enter Tutorial.

3  Click Create New Package.

   You see the Package Properties window.

4  Click OK.

   When you click on the Installed Packages expansion sign, it now includes Tutorial.

Define and install a new component

❖ Using Jaguar Manager, define a new component for the package you just created:

   Click on the Tutorial package.

   5  Select File | Install Component.

   6  In the Component wizard, select Create and Install a New Component.

      For the component name, enter *CPPArithmetic*.

   7  Click Create New Component.

   8  Select the Define New Component check box. Click OK.

      You see the Component Properties window.

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

      | Field            | Value                      |
      |------------------|----------------------------|
      | Description      | Tutorial C++ component     |
      | Codeset          | Server’s Codeset (default) |
      | IDL Interface    | Tutorial::CPPArithmetic   |
      | Component Type   | C++                        |
Define the multiply method

After the component is defined, define methods for it that the client will call:

Expand the Tutorial package. Select the CPPArithmetic component.

12 Select File | New Method.

Assign the name multiply to the method.

13 Click Create New Method.

You see the Method Properties window.

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

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.

14 Click OK to close the New Parameter dialog box.

Repeat steps 6 and 7 to add a second parameter named m2 and with a type of double.

15 Click OK to close the Method Properties dialog box.

16 At this point, Jaguar Manager has created an IDL interface for the component with a multiply method. You can view the generated IDL interface as follows:

a Expand the Modules folder in Jaguar Manager.

Right-click the icon for the Tutorial module.

b Choose Edit Module IDL ... from the pop-up menu. The module’s definition displays in the IDL Editor window.
c View the IDL definitions for the module, interface, and method that you created earlier.

d Close the IDL Editor window by choosing File | Exit from the menu.

For more information on defining methods and parameters, refer to Chapter 4, “Defining Component Interfaces” in the Jaguar CTS Programmer’s Guide.

Generate stubs, skeletons, and implementation templates

Once you have created the package, component, and methods, you need to generate the stub files for the component. The client-side applet uses the stubs to invoke the server-side component methods.

❖ To generate the stubs and skeletons
Click on the Tutorial package and select the CPPArithmetic component.

17 Select File | Generate Stubs/Skeletons.
Select Generate Stubs, then check C++ Stubs. In the C/C++ Code Base field, enter the path to the Jaguar include subdirectory—for example: %JAGUAR%\include.

Unselect Generate Java Stubs.
Select Generate Skeletons.
Under Skeletons in the C/C++ Code Base field, type the full path to a subdirectory where you will create the component source files—for example:

d:\jagtut

18 Click Generate.

Write the server-side code

❖ To write the server-side code:
Navigate to the directory that you specified for C/C++ Code Base in the skeleton generation options, 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 (Visual C++) makefile.

Rename the implementation files to *CPPArithmeticImpl.hpp* and *CPPArithmeticImpl.cpp* (delete the .new extension).

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:

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

19 Save your changes.

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

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

Change the ODBCHOME definition to match the directory where you have installed Microsoft Visual C++. Save your changes.

Build the DLL for the component by running *nmake* (no arguments are required). You should see a new file called *libCPPArithmetic.dll*. Copy this file to the Jaguar *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*. 

19 Save your changes.

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

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

Change the ODBCHOME definition to match the directory where you have installed Microsoft Visual C++. Save your changes.

Build the DLL for the component by running *nmake* (no arguments are required). You should see a new file called *libCPPArithmetic.dll*. Copy this file to the Jaguar *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*. 

19 Save your changes.
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` subdirectory of your Jaguar installation. Here is the source for `arith.cpp`:

```c++
/*
** arith.cpp -- Example C++ client for the Jaguar C++
** tutorial.
**
** This program connects to the Jaguar naming server,
** creates an instance of the Tutorial/CPPArithmetic
** component, and invokes the multiply method.
**
** Usage:
** arith [ -
** ORBNameServiceURL iiop://<host>:<port>/<context> ]
**
** The ORBNameServiceURL must be specified unless you specify
** the URL by setting the JAG_NAMESERVICEURL environment
** variable. The URL components are as follows:
**
** <host> is the Jaguar naming server host name.
**
** <port> is the naming server’s IIOP port (9000 in the
** default configuration.
**
** <initial-context> is the naming server’s initial
** context, for example, US/Sybase/Jaguar/Pubs.
**    In the default configuration, the context
**    is an empty string.
**
** If you are using the default configuration, your
** component’s name server is the Jaguar server where it is
** installed, and the initial context is null. In this case, enter the URL as
*/
```
#include <stdio.h>
#include <iostream.h>
#include <string.h>
#include <Jaguar.hpp>
#include <SessionManager.hpp>
#include <CosNaming.hpp>
#include <Tutorial.hpp>  // Stubs for interfaces in Tutorial

int main(int argc, char** argv)
{
    const char *usage =
    "Usage:\n	-arithmetic \n-ORBNameServiceURL iiop://<host>:<port>/<initial-context>\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 {

        cout << "Creating Jaguar session\n\n";

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

        // Obtain the CORBA CosNaming initial naming context
        // that we will use to resolve objects by name
        // The ORB retrieves the naming server address from
        // command line arguments or the environment.
        CORBA::Object_var obj =
>resolve_initial_references("NameService");
    CosNaming::NamingContext_var nc =
        CosNaming::NamingContext::_narrow(obj);
    if (CORBA::is_nil(nc)) {
        cout << "Error: Null NamingContext instance. Exiting.";
        return -1;
    }
    
    // Build a CosNaming::Name object that contains the
    // name of the tutorial component, Tutorial/CP
    PARithmetic
        name[0].id = CORBA::string_dup( component_name );
    name[0].kind = CORBA::string_dup( "" );
    
    // Obtain a factory for component instances by
    // resolving the component name
    cout <<
    "Creating component instance for "
    << component_name << 
    \n";
    obj = nc->resolve(name);
    SessionManager::Factory_var arithFactory =
        SessionManager::Factory::_narrow(obj);
    
    if (CORBA::is_nil(arithFactory)) {
        cout << "ERROR: Null component factory. "
            << tutorial_help ;
        return -1;
    }
    
    // Use the factory to create an instance, passing
    // username and password for authorization
    Tutorial::CPPArithmetic_var arith =
        Tutorial::CPPArithmetic::_narrow(
            arithFactory->create("Guest", "GuestPassword") );
    
    // Verify that we really have an instance.
    if (CORBA::is_nil(arith)) {
        cout << "ERROR: Null component instance. "
            << tutorial_help ;
// 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 a
ny other
// CORBA system exception.

// Requested object (component) does not exist.
catch (CosNaming::NamingContext::NotFound &nf)
{
    cout << "Error: Component " << component_name
    << " not found. "
    << "Verify that the component has been cr
eated "
    << "properly in Jaguar Manager. "
    << "Also check the server log file for ad
ditional "
    << "information.";
}

// Authentication or authorization failure.
catch (CORBA::NO_PERMISSION npe )
{
    cout << "Error: CORBA:: NO_PERMISSION exceptio
n. "
    << "Check whether login authentication is enabled "
    << "for your server and whether the component has "
    << "restricted access."
// 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 Jaguar host
        "
        << "and IIOP port number are correct and
        "
        << "that the server is running.\n"
        << usage;
}

// 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.\n";
}

// Anything else.
catch ( CORBA::OBJ_ADAPTER )
{
    cout << "Error: CORBA OBJ_ADAPTER \n";
}
catch ( CORBA::SystemException cse )
{
    cout << "Error: CORBA System Exception. Check that "
        "the Jaguar hostname and IIOP port are
        "
        << "specified correctly, and check the se
server's' " << " error log for more information.\n" << usage;
}
return 0;
}

Compile the client executable

To compile arith.cpp, on Windows, run this batch file:

SETLOCAL
set INCLUDE=.;%JAGUAR%\include;%INCLUDE%
set LIB=%JAGUAR%\lib;%LIB%
c /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

Run the client executable

If you have not restarted the Jaguar server since creating the CPPArithmetic component, do so now before running the client program.

Classpath conflicts on development workstations
If you are running the sample client on a machine where EAServer has been installed, you must configure your browser to eliminate conflicts between Jaguar classes that are downloaded with the applet and classes that are loaded from the local classpath. See “Configuring browsers to run applets on development machines” on page 30 for instructions.

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

arith -ORBNameServiceURL iiop://host:iiop-port

For example:

arith -ORBNameServiceURL 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.
Tutorial: ActiveX Components

About this chapter

In this tutorial, you will create a simple application involving an ActiveX component and a Java applet client that calls a method in the component.

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of the sample application</td>
<td>50</td>
</tr>
<tr>
<td>Tutorial requirements</td>
<td>51</td>
</tr>
<tr>
<td>Creating the application</td>
<td>52</td>
</tr>
</tbody>
</table>
Overview of the sample application

The application performs the following steps:

1. The client-side applet, developed with Java, instantiates the middle-tier ActiveX component, AXArithmeticCtl.
2. The applet calls the multiply method in AXArithmeticCtl.
3. The multiply method computes the product of the input values, then returns the result.
4. The applet displays the result for the end user.
Tutorial requirements

To create this tutorial, you need:

- The Jaguar CTS software
  
  For installation instructions, see the Enterprise Application Server Installation Guide.

- ActiveX development environment
  
  For this tutorial, in addition to the applet, you will also create a server-side component that runs in the Jaguar server. This component performs a simple activity in the middle tier when invoked by the applet running in the browser. You can develop ActiveX components using any popular development environment, including:
  
  - PowerBuilder
  - Microsoft Visual C++

- Java development environment
  
  Jaguar comes with two versions of Sun Microsystems Java Developer’s Kit: JDK 1.0.2 and JDK 1.1x, where x is the latest JDK release that has been tested with Jaguar. “Java compiler scripts” on page 20 describes script files that will set the environment to run the JDK compilers that are included in your Jaguar installation.

  In the sample application, you will be developing a Java applet that will handle user interaction. For the applet development, you may use any of the off-the-shelf Java rapid application development (RAD) tools including:

  - Sybase Power J
  - Symantec Visual Cafe
  - Microsoft Visual J++

- To run the tutorial applet, you must have a Web browser that is compatible with JDK 1.1. Netscape 4.0, Internet Explorer 4.0, or Sun’s HotJava browser have all been tested with this tutorial.

Note

Older versions of the Netscape and Microsoft browsers support only JDK 1.0.2. You may need to create clients that are compatible with JDK 1.0.2 to satisfy your site’s deployment requirements. See the Jaguar CTS Programmer’s Guide for more information.
Creating the application

To create the sample application:

1. Write the ActiveX server-side component using Power++
2. Build, register, and deploy the component DLL
3. Verify that the component is registered
4. Start the Jaguar server and Jaguar Manager
5. Define a Jaguar package and import the component definition
6. Generate stubs using Jaguar Manager
7. Compile the stub code
8. Write the client-side code
9. Create the HTML page and run the sample

Write the ActiveX server-side component using Power++

This section shows you how to create the server-side component. The ActiveX component (AXArithmetic.dll) is included in your Jaguar installation. If you do not need or want to build the component, you can skip to “Verify that the component is registered” on page 54.

❖ Create the server-side component:

1. Start Power++.
2. Select File | New | Target.
3. Select ActiveX Server DLL as the target type and click Next.
4. Assign it a target name of AXArithmetic and place it in a directory called c:\AXArithmetic. Select Finish.
5. Select File | Save Project. Accept the default values.
6. In the left panel of the AXArithmetic - Classes window, select the AXArithmetic.dll | IAXArithmetic folder.
7. Select Class | Insert | Method.

In the Method wizard, assign the method the name Multiply. In the prototype, click between the parentheses, and add this text to define the method parameters:
double m1, double m2, double *result

**Note**
Since the last parameter is a pointer to `double`, Power++ defines an `IDispatch` interface in which the method returns `double`. After the component is imported into Jaguar, the Jaguar Manager definition of the `multiply` method will return `double`.

8 Click Finish. Power++ displays a window with a cursor in the `IAXArithmetic::multiply` subroutine. Insert the following code before the `return` statement:

```c
*result = m1 * m2;
```

9 Select File | Save Project.

At this point, you are ready to build the component DLL. For more information on defining ActiveX components in Power++, see the *Power++ Component Creation Guide*.

---

**Build, register, and deploy the component DLL**

When you are creating ActiveX DLLs, Power++ allows you to configure the Run command to build, register, and deploy the component DLL.

The steps below configure the Run menu so that the component DLL is registered, then copied, to the Jaguar `dll` directory.

❖ **Build and deploy the component DLL:**

1 Select Run | Run Options from the Power++ menu.

2 In the General tab of the Run Options window, make sure Do nothing is selected.

3 Under Copy Files:
   a Select the option to copy the DLL. Use the Browse button to select the Jaguar `dll` directory as the destination folder.
   b Select the option to register the ActiveX server automatically.
   c Do not select the option to reinstall the component in the Component Palette.

4 Click OK to close the Run Options window.

5 Select File | Save Project.
6 Select Run | Run from the Power++ menu.
Verify that AXArithmetic.dll has been built in the c:\AXArithmetic\debug directory and copied to the Jaguar dll directory.

Verify that the component is registered
ActiveX component DLLs must be registered in the NT Registry before applications can call the DLLs. If you configured the Run options correctly, Power++ should have automatically registered your component.

In Jaguar, the component’s name in Jaguar Manager must match the component’s program ID (progid) that is stored in the NT Registry. In the steps below, you will confirm the component registration.

❖ Verify that the component is registered:
1 Open the Registry by selecting Start | Run. Enter regedt32 in the window. Click OK.
2 Select the HKEY_CLASSES_ROOT on Local Machine window.
3 Scroll down to AXArithmeticCtl. The AXArithmetic ActiveX control has been registered to your machine under this name. Use this name as the component name for the ActiveX control in Jaguar Manager. There may also be an AXArithmeticCtl.1 entry, which you can ignore.
4 Close regedt32.

Start the Jaguar server and Jaguar Manager
❖ Start the Jaguar server:
• If the Jaguar server is not already running, follow the instructions under “Starting Jaguar” on page 6 to start the server.
❖ Start Jaguar Manager:
• If Jaguar Manager is not already running, start it as described in “Starting Jaguar Manager and Security Manager” on page 12.
Define a Jaguar package and import the component definition

This section shows you how to create a package for a new component, and import a component’s method definition into the package. You can either import ActiveX method definitions from a type library (.tlb) file or a dynamic link library (.dll) file into Jaguar Manager, or you can manually enter method definitions. This tutorial imports method definitions from a type library file.

For more information about importing or creating method definitions and configuring packages and component properties, see the Jaguar CTS Programmer’s Guide.

Define a new package

Packages provide a way of organizing related components in a Jaguar application. All components created in the Jaguar tutorials are installed in the Tutorial package. Follow these steps to create the Tutorial package if it does not already exist:

❖ Create the Tutorial package:

1. Navigate to:
   Jaguar Manager | Servers | Jaguar

2. In the right panel, click Packages. If the Tutorial package is displayed, skip to the next section, “Define the Jaguar ActiveX component” on page 55.

3. 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.

   When you click on the Installed Packages expansion sign, it now includes Tutorial.

Define the Jaguar ActiveX component

You must install the AXArithmetic component on the Jaguar server before you can add methods to it.
Define the Jaguar ActiveX component:
1. Click on the Tutorial package.
2. Select File | Install component.
3. Click on Create and Install a New Component.
   For the component name, enter AXArithmeticCtl. Click Create New Component.
4. Select Import from ActiveX File. Click OK.
5. In the Import ActiveX window, click Browse. Navigate to the directory called c:\AXArithmetic. Change to the release subdirectory, then double-click the file DTAXArithmetic.tlb.
6. On the Component Properties sheet, select ActiveX. Set the stub package name to Tutorial. Click OK.
7. You should see AXArithmeticCtl displayed as a component in the Tutorial package. Select the AXArithmeticCtl component. Its methods display on the right side of the screen. Double-click on the multiply method to display its property sheet. Verify that the method returns double and accepts two double input parameters.

Generate stubs using Jaguar Manager

Once you have created the package, component, and methods, you need to generate the stub code used as a basis for creating the client-side applet that invokes the server-side component methods.

Generate stubs:
1. Click on the Tutorial package and select the AXArithmeticCtl component.
2. Select File | Generate Stubs/Skeletons.
3. Select Generate Stubs, then select Generate Java Stubs. Fill in the Java Stubs fields as follows:
   a. Select Using CORBA datatypes from the drop-down list.
   b. Select JDK 1.1 for Java version.
   c. In the Java Code Base field, type the full path to your Jaguar html\classes subdirectory—for example, %JAGUAR%\html\classes.
   d. For Java Package, enter Tutorial.
4 Unselect Generate C++ Stubs.
5 Unselect Generate Skeletons.
6 Click Generate.

Compile the stub code

Change to the Jaguar html\classes\Tutorial subdirectory and compile the generated files using a JDK 1.1 compiler—for example:

```
    cd %JAGUAR%\html\classes\Tutorial
    d:\j11 *.java
```

Write the client-side code

Create a directory called %JAGUAR%\html\classes\TutorialApps if it does not exist. In this directory, create the Java file below as AXClient.java.

A copy of AXClient.java can be found in the html/docs/tutorial subdirectory of your Jaguar installation. Here is the source for AXClient.java:

```
//
// This is a sample client that invokes the
// AXArithmeticCtl component created in the Jaguar
// ActiveX component tutorial.
//
package TutorialApps;

import org.omg.CORBA.*;
import org.omg.CosNaming.*;   // CORBA CosNaming interfaces
                         // CosNaming user exceptions
import SessionManager.*;
import java.awt.*;

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

public class AXClient extends java.applet.Applet {

    // User interface controls
```
Button mult_button;
TextField ml_text;
TextField m2_text;
TextField result_text;

// Component’s name (relative to the server’s
// initial name context), of the form package/compon
ent
String _compName = "Tutorial/AXArithmeticCtl";

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

public void init()
{

    // Draw GUI controls
    ml_text = new TextField("  ");
    ml_text.setText("2.5");
    this.add(ml_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);

    try {
        // Initialize the CORBA client-side ORB and
        // obtain a stub for the Jaguar component instan
ce.
        //
        System.out.println("... Creating Jaguar session.
        ");

        // Initialize the ORB. Note that the org.omg.COR
BA.ORBClass
Chapter 4  Tutorial: ActiveX Components

// property must be set in applet parameters.
//
java.util.Properties props = new java.util.Properties();
ORB orb = ORB.init(this, props);

NamingContext nc = null;
org.omg.CORBA.Object objRef = null;

// Create an instance of the Jaguar SessionManager
// CORBA IDL object.
//
SessionManager.Manager manager = null;

NameComponent managerName[] =
{ new NameComponent("AuthenticationService", "") }
manager = ManagerHelper.narrow( nc.resolve(managerName) );

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

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

// Create a stub object instance for the
// Tutorial/AXArithmeticCtl Jaguar component.
//
NameComponent compNc[] =
{ new NameComponent(_compName, "") }
Factory compFactory =
FactoryHelper.narrow ( nc.resolve(compNc) )
```java
// _comp = Tutorial.AXArithmeticCtlHelper.narrow(
   compFactory.create(session) );

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

} catch (NotFound nfe) {
   // This can happen if you have installed
   // the tutorial component under a different
   // package or component name in Jaguar Manager
   System.out.println("Error: Component "+ _comp
   Name
   + " not found. Check the package and compone
   nt "+ "name in Jaguar Manager.\n"
   System.out.println("NotFound exception details
   :\n")
   nfe.printStackTrace();
   _comp = null;
} catch (org.omg.CORBA.UserException ue) {
   // Check for other CosNaming exceptions
   System.out.println("CORBA CosNaming exception:
   \n"
   + ue.toString());
   ue.printStackTrace();
   _comp = null;
} catch (org.omg.CORBA.SystemException se) {
   System.out.println("Received CORBA system exception 
   +"while instantiating component:\n"
   + se.toString() );
   se.printStackTrace();
   _comp = null;
}

} // 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.printStackTrace();
    }

} // doMultiply

Compile this application using a JDK 1.1 compiler:

javaj1 AXClient.java
Create the HTML page and run the sample

In a text editor, create an HTML file that calls the applet. Name the file `runaxtut.html` and create it in the `%JAGUAR%html\classes\TutorialApps` subdirectory.

A copy of this file is located in your Jaguar installation directory, in the `html/docs/tutorial` subdirectory. As an alternative to creating the file, copy this file to your `%JAGUAR%html\classes\TutorialApps` subdirectory. Here is the text of `runaxtut.html`:

```html
<html><body bgcolor="#FFFFFF">
<head><title>This Applet runs the Jaguar tutorial ActiveX component.</title></head>
<center>
<applet codebase="/classes" code="TutorialApps/AXClient.class" width=600 height=400>
<h2>This would be a Cool Applet, but you are not running a Java enabled browser...
</h2>
</applet>
</center>
<hr>

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 Jaguar, edit the port number in the HTML file to match.
Chapter 4   Tutorial: ActiveX Components

Run the sample

You do not need to restart the Jaguar server before running the sample.

Classpath conflicts on development workstations
If you are running the sample client on a machine where EAServer has been installed, you must configure your browser to eliminate conflicts between Jaguar classes that are downloaded with the applet and classes that are loaded from the local classpath. See “Configuring browsers to run applets on development machines” on page 30 for instructions.

❖ Run the sample:
1 Start a Netscape 4.x, Internet Explorer 4.x, or HotJava Web browser. If necessary, you can run the applet on a different machine than the Jaguar 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 Jaguar server’s host name for host:

   http://host:8080/classes/TutorialApps/runaxtut.html

   As installed, Jaguar 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 Jaguar method. The return value (product) is displayed.

Debugging

❖ Debug if necessary:

• Look at the %JAGUAR%\bin\srv.log file and the Java console to check debug messages.
In this tutorial, you will run an applet that uses SSL security features supported by Jaguar.

**Table of Contents**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of the security tutorial</td>
<td>66</td>
</tr>
<tr>
<td>Tutorial requirements</td>
<td>67</td>
</tr>
<tr>
<td>Installing certificates in Netscape</td>
<td>68</td>
</tr>
<tr>
<td>Creating a user certificate</td>
<td>74</td>
</tr>
<tr>
<td>Creating and assigning a security profile to a listener</td>
<td>75</td>
</tr>
<tr>
<td>Running the SSL sample applet</td>
<td>78</td>
</tr>
<tr>
<td>Debugging the SSL sample applet</td>
<td>79</td>
</tr>
</tbody>
</table>
Overview of the security tutorial

You should be familiar with SSL concepts and terms before you run this tutorial. Refer to the *Jaguar CTS System Administration Guide* for an overview of SSL concepts and how to use Security Manager.

In this tutorial, you will:

1. Use Netscape’s security features to:
   - Request and install a personal certificate in your Netscape browser.
   - Load Jaguar’s test Certificate Authority (CA) certificate in Netscape and mark it “trusted.” Once you have marked the certificate trusted, Netscape trusts the identity of the Jaguar server who presents a certificate signed by the Jaguar test CA.

2. Use Security Manager to:
   - Load the signer’s certificate (the CA that signed your personal certificate) in Jaguar and mark it trusted. This is required so that Jaguar trusts the identity of the client who presents a certificate signed by the signing CA.
   - Generate a certificate signed by the test CA. Once you have assigned this certificate to a listener, it authenticates Jaguar when you connect to the listener from your browser.
   - Create a security profile and assign it to a listener. This establishes the security parameters of your HTTPS and IIOPS ports.

3. Connect to the HTML page that contains the applet from your browser on a secure HTTPS listener and run the sample applet.
Tutorial requirements

To run the tutorial, you need:

- The Jaguar CTS software
  
  For installation instructions, see the Enterprise Application Server Installation Guide.

- A Netscape 4.05 Web browser.

---

**Note**

Other browsers that support digital certificates may work, but have not been tested with this tutorial.
Installing certificates in Netscape

To install Jaguar’s test CA and a personal certificate in your Netscape browser:

1. Start the Jaguar server, Jaguar Manager, and Security Manager
2. Export the test CA
3. Log in to the Netscape token and load the test CA
4. Obtain and install a personal certificate

Start the Jaguar server, Jaguar Manager, and Security Manager

❖ Start the Jaguar server:
   • If the Jaguar server is not already running, follow the instructions under “Starting Jaguar” on page 6 to start the server.

❖ Start Jaguar Manager and Security Manager:
   • If Jaguar Manager and Security Manager are not already running, start them as described in “Starting Jaguar Manager and Security Manager” on page 11.

Export the test CA

You need to export the test CA signing authority from Jaguar, and then load it in your browser and mark it as trusted. This prevents Netscape from displaying warnings about untrusted certificate authorities when you use server authentication with Jaguar listeners.

❖ Export the test CA from Security Manager:
   1. Select the CA Certificates folder.
   2. From the right side of the window, highlight the Sybase Jaguar User Test CA.

      If the Sybase Jaguar User Test CA does not appear, you must first create it. Refer to “Creating a test CA” in the Jaguar CTS System Administration Guide for instructions.

4 Select the format type for the exported certificate. Select “Binary encoded X509 Certificate.”

5 Select Save to File and enter the full path to a file that will contain the test CA’s certificate.

Do not add any extension to the file name. A .crt extension is automatically appended to the exported file by Security Manager. Netscape 4.05 recognizes this extension as a certificate and handles it accordingly.

6 Click Finish. The certificate is exported to the file you selected.

Refer to “Test CA management” in the Jaguar CTS System Administration Guide for more information about the test CA.

Log in to the Netscape token and load the test CA

The Netscape token manages the certificates and private keys in your browser. Before you can load the test CA in Netscape you must be logged in to the Netscape token.

Log in to the Netscape token

❖ Log in to the Netscape token from your browser:

1 Click the Security icon from the tool bar.

2 Click Cryptographic Modules on the left side of the window.

3 Click Netscape Internal PKCS #11 Module in the Cryptographic Modules window.

4 Click the View/Edit button.

5 Click Communicator Certificate DB in the Edit Security Module window.

6 Click the Login button. You must provide the token’s login password.

To verify the state of the token, click the More Info button. The Token/Slot Information window displays information about the Netscape token. Verify that the State of the token is Ready.
Note
If you do not have the password or have forgotten it, close Netscape. Find the file key3.db in your user profile directory and rename the file. Restart Netscape and you should be prompted to establish a new password. You may need to perform steps one through five then select Initialize token to establish a new password.

7 Once logged in to the Netscape token, click OK on the open dialog boxes until you return to the main Netscape window.

Load the test CA

Load the test CA in your browser:

1 Enter the full path of the exported test CA file in Netscape’s Netsite field. For example:
   c:test_ca.crt

2 Select “Open it” from the dialog box and click OK. Netscape recognizes the .crt extension as belonging to a CA and displays a series of dialog boxes asking if you want to accept the CA.

3 Accept the defaults on the dialog boxes.

4 If prompted, select Accept this Certificate Authority for Certifying Network Sites. This allows Netscape to connect to Jaguar ports that require authentication and accepts the certificates signed by the test CA without displaying warning dialog boxes.

5 You may be prompted to enter the nickname of the test CA. Enter a name, (for example, Sybase Test CA) then click Finish. The nickname identifies the test CA within the Netscape token.

Verify that the test CA has been loaded:

1 Click the Security icon on the tool bar.

2 Click Signers on the left side of the window to display a list of the CAs that Netscape accepts.

3 Scroll down and select Sybase Test CA. You can edit information, verify that the test CA is valid, or delete the test CA from Netscape by using the corresponding buttons in the Certificate Signers’ Certificate window.
Obtain and install a personal certificate

You need a personal certificate installed in your browser before the sample applets can attach to Jaguar listener ports that require client authentication.

Before you begin

There are a variety of ways to get a personal certificate:

- **Use the sample certificates**  
  Jaguar comes with two sample certificates signed by the test CA that you can use to authenticate yourself when connecting to secure Jaguar listeners. To use the sample certificates, you must install and log in to the Sybase PKCS #11 module, instead of the Netscape PKCS #11 module.

  Refer to “Installing Sybase PKCS #11 into Netscape 4.0x” in the *Jaguar CTS System Administration Guide* for more information about installing and logging in to the Sybase PKCS #11 module.

- **Attach to an in-house CA**  
  Supply the required information to request a personal certificate.

  For the purpose of this tutorial, an in-house Netscape CA was used to obtain a personal certificate. Refer to “Obtaining a certificate from a Netscape certificate server” on page 77 for detailed instructions.

- **Use a public CA**  
  You can obtain your certificate from any public CA. A number of public CAs are available through your Netscape browser. To request a certificate:

  a  
  Click the Security icon on the tool bar.

  b  
  Click Yours on the left side of the window. This displays a list of your certificates.

  c  
  If no certificates are displayed, you need to get one. Click Get a Certificate. You see a Web page of public CAs.

  You need to obtain a certificate from a CA that Jaguar recognizes, or use Security Manager to install the CA’s certificate and mark it trusted. In Security Manager, click the Trusted CAs folder to display a list of the trusted certificate signers that Jaguar recognizes.

  d  
  Select a CA and follow the instructions to obtain your certificate.

Obtain a certificate from a Netscape certificate server

- **Obtain a certificate from a Netscape certificate server:**

  1  
  Connect to the certificate server URL.
2 Click Request a Personal Certificate from the Public Menu.
3 Supply the information in the Request a Personal Certificate form.
4 Select 1024 (High Grade) encryption from the drop-down list.
5 Click Submit Request. Netscape generates the public and private key pair and forwards your request to the CA. Once approved, the CA will send e-mail instructions to you on how to pick up the certificate.
6 Connect to the URL indicated in the e-mail message. Follow the instructions to install the certificate. You must install the certificate on the same machine that generated the request, since this is where the corresponding private key resides.

❖ View your certificate:
   1 Click the Security icon on the tool bar.
   2 Click Yours on the left side of the window. This displays a list of your certificates that are loaded in Netscape. You should see your newly imported certificate in this list.

Install the CA’s certificate in Jaguar

You need to install the certificate of the signer (CA) of your personal certificate in Jaguar. Otherwise, when you offer your personal certificate to Jaguar for client authentication, Jaguar does not recognize it as being signed by a trusted CA.

❖ Install the CA’s certificate in Jaguar and mark it as trusted:
   1 In Netscape, go to the URL of the CA that issued your certificate.
   2 Locate and copy the signer’s certificate. For the Netscape certificate server:
      a Click List Certificates.
      b Click Run Query.
      c Highlight the certificate, including the BEGIN through the END lines.
      d Click Details in the Issuers certificate field.
      e Select Edit | Copy.
   3 In Security Manager, double-click the Security Manager icon.
   4 Highlight the CA Certificates folder.
5 Select File | Install Certificate.
6 Paste the signer’s certificate into the window provided.
   If you have copied the certificate to a file, click Import from File and enter
   the name of the file.
7 Click Install.
   The CA’s certificate is displayed when you highlight the CA certificates folder.

❖ Mark the certificate trusted:
1 Highlight the certificate.
2 Select File | Certificate Info.
3 Click the Trusted Certificate box.
4 Click Done.
   The CA’s certificate is displayed when you highlight the Trusted CAs folder.
   Restart the Jaguar server. Certificates signed by the installed CA are now
   accepted for client authentication by Jaguar.
Creating a user certificate

In this section you will create a user certificate that is signed by the test CA. Later, you will assign this certificate to a security profile, and assign the security profile to a listener. The listener uses the certificate to authenticate Jaguar.

❖ Create a user certificate from Security Manager:

1. Highlight the CA Certificates folder.
2. Select File | Generate User Test Certificate.
3. Provide the information in the Generate User Test Certificate wizard as follows:
   - **Key Strength**  Select 512 from the drop-down list.
   - **Validity Period**  Select two months from the drop-down list. The validity period determines how long the certificate is valid. When Jaguar authenticates itself using this certificate, Netscape examines the validity period to see if it has expired.
   - **Key Label**  Enter Tutorial_cert for the name that identifies the certificate.
   - **SSL Server**  Select this box since you will use this certificate to authenticate Jaguar.
   - **SSL Client**  The same certificate can also be used by clients for authentication. Since this certificate will not be used to authenticate the client, do not select this box.
   - **Mark Private Key as Exportable**  Since you are not using this certificate on other systems, do not check this box.
4. Click Next. Provide your personal and site information as requested in the Certificate Request Information window. Refer to “User test certificate information” in the Jaguar CTS System Administration Guide for information on these fields.
5. Click Finish. Security Manager generates a user certificate that is signed by the test CA. To view the certificate, highlight the Users Certificates folder.
Creating and assigning a security profile to a listener

In this section you will define a new security profile, which includes a security characteristic. The security characteristic determines characteristics of the client-Jaguar connection, such as:

- **Authentication**  The security profile you create for this tutorial requires certificates for authentication from both the client and server.
- **Encryption**  The strength and method of encryption. The security profile you create for this tutorial will not encrypt data.

❖ Create a security profile:
1. Double-click the Jaguar Manager icon.
2. Click the Security Profiles folder.
4. Enter *user_test* as the name of the security profile and click Create New Security Profile.
5. Enter the information in the SSL tab of the Security Profile Properties window as follows:
   - **Description**  Enter *sample security profile* as the description of this security profile.
   - **Use Entrus**  Uncheck this box. You would check this box if you were using an Entrust ID for authentication.
   - **Security Characteristic**  Select *sybpks_intl_mutual_auth* from the drop-down list. A description of this security characteristic displays in the Description window.

You have access to only the international/export security characteristics unless you run the upgrd128 upgrade. Refer to “Upgrading to stronger encryption” in the *Jaguar CTS System Administration Guide* for more information.

Refer to “Security characteristics” in the *Jaguar CTS System Administration Guide* for more information about security characteristics.

- **Certificate Label**  Select *Tutorial_cert* from the drop-down list. This is the label of the certificate you created earlier. The security profile uses this certificate to authenticate Jaguar. If you have not logged in to Security Manager, you are prompted for a PIN.
• **PIN**  Enter the password (PIN) and press ENTER. This is the same PIN that allows access to Security Manager. The default PIN is `sybase`. If you have changed this PIN, enter the new PIN. Refer to “Changing the user PIN” on page 102 for more information.

• Click Save. Jaguar Manager displays the new security profile.

You can now assign the user_test security profile to a listener. Refer to “Security profiles” on page 120 for more information about security profiles.

---

**Assign a security profile to a listener**

A listener identifies Jaguar ports that accepts connection requests from clients using the following protocols:

- HTTP
- HTTPS
- IIOP
- IIOPS
- TDS

When you define a listener, you choose a port number, the protocol, and, for secure protocols IIOPS and HTTPS, assign a security profile.

❖ **Assign the test_profile security profile to a listener:**

1. Double-click the Jaguar Manager icon.
2. Double-click the Servers folder.
3. Double-click the Jaguar icon.
4. Click the Listeners folder.
5. Select File | New Listener.
6. Enter `https3` for the listener name and click Create New Listener.
7. When you see the Listener info window, supply the following:
   - **Protocol**  Select HTTPS from the drop-down list. You will use HTTPS as the protocol to retrieve the HTML page that contains the sample applet.
   - **Host**    Enter the name of the Jaguar server host.
   - **Port**    Enter the port number on the host machine for this listener. If not in use by any other service, enter `8083`. 

---
- **Jaguar Security Profile** Select the user_test security profile from the drop-down list.

8 Click Save.

9 Restart Jaguar:
   a Highlight the server to which this listener belongs.
   b Select File | Restart.

You now have a Jaguar listener that accepts HTTPS connection requests at port 8083 and requires client and Jaguar authentication.

Refer to “Listeners” in the *Jaguar CTS System Administration Guide* for more information about listeners.
Running the SSL sample applet

The SSL sample applet contains code for both a Java and a C++ server component. The applet instantiates and runs the JUserCredentialTest (Java) or CUserCredentialTest (C++) component. The component retrieves and the applet displays information about the client certificate.

Complete instructions for running the applet are in the file %JAGUAR%/html/classes/Sample\SecurityDemo\readme.html.

❖ Follow the instructions in the readme.html file to do the following:

1 Import the SecurityDemo package into Jaguar Manager.

   The SecurityDemo package contains a Java and a C++ component. These components both implement the SecurityDemo::UserCredentialTest interface.

2 Generate stubs and skeletons.

   You need to generate the stub files for the Java applet and the skeleton files for the server component.

3 Compile the Java source files.

4 Run the applet.

   You are instructed to load the HTML page that contains the applet at port 8080. If you connect to port 8080, authentication requirements are determined by the IIOP listener to which the applet connects.

   You can also load the HTML page by connecting to the HTTPS listener port 8083 that you created earlier. Before the browser loads the page, you need to accept Jaguar’s certificate and supply a personal certificate for client authentication.

   The SSL sample applet connects to the preconfigured IIOP listeners, IIOP at port 9000, IIOPS at port 9001, and IIOPS at port 9002. For the applet to run successfully, verify that the host name for these listeners is the same as the host name for the HTTPS listener (8083). Refer to “Preconfigured listeners” in the Jaguar CTS System Administration Guide for more information.
Debugging the SSL sample applet

If you have difficulty running the sample:

- View the *srv.log* file to verify that the listeners are running. Refer to “Preconfigured listeners” in the *Jaguar CTS System Administration Guide* for information.

- Check the Java console in your browser for error messages. To view the console, select Communicator | Java Console.

- If the *srv.log* or Java console indicates an untrusted certificate error, make sure you have loaded the test CA’s certificate from Security Manager in Netscape, and that you have installed the signer’s certificate (of your personal certificate) in Security Manager.

- Make sure that the listener’s Hostname field for all preconfigured listeners and the listener you created for this tutorial are set to the actual name or IP address of the host and *not* localhost.
Index

A

ActiveX
devision environments 51
ActiveX tutorial
ActiveX server-side component 52
cient-side applet 57
debugging 63
generating stubs for 56
HTML page for 62
running from Web browser 63
server-side code 52
add parameter begin step 23
add parameter end step 23

D

debugging
the ActiveX tutorial 63
the Java tutorial 31
defining
packages, components, and methods for C++ tutorial 36
packages, components, and methods for Java tutorial 21
Development environments
ActiveX 51
development environments
Java 19, 51

C

C++ tutorial
cient-side applet 41
defining packages, components, and methods for 36
generating stubs for 39
HTML page for 46
server-side code 39
component
definition 2
executing methods on 4
instantiating 4
component models
supported 2
concepts 3
connecting to Jaguar
using Profile Manager 13
connection screen
host name 12
jagadmin password 12
port number 12

E

environment variables
INCLUDE 15
LIB 15

H

HTML page
for ActiveX tutorial 62
for C++ tutorial 46
for Java tutorial 29

I

INCLUDE
environment variable 15
instantiating
components 4
Index

J
jagadmin
  connecting to Jaguar Manager 12
Jaguar Manager 11
  and the Jaguar server 11
  configuring 11
  running as an application 12
Jaguar server
  starting 6
  starting a user-defined server 6
JAGUAR_RANDOMSEED 10
Java
  development environments 19, 51
Java tutorial
  client-side applet 25
  debugging 31
  defining packages, components, and methods for 21
  generating stubs for 23
  HTML page for 29
  running from Web browser 30
  server-side code 24
JDK
  batch file 20

L
LIB
  environment variable 15
listeners
  verifying 15

M
multitier
  application development overview 3

N
NT
  registering your components 54

O
overview
  multitier application development 3

P
package
  definition 2
port settings
  verifying HTTP, IIOP, and TDS 15
Power++
  creating an ActiveX component 52
Profile Manager
  connecting to Jaguar 13
proxy objects
  and stubs 2

R
RANDOMSEED variable 10
regdt32
  registering components with 54
  registering components on NT 54
running Jaguar Manager
  as an application 12

S
sample
  Jaguar application 16
session
  definition 3
skeleton
  definition 3
starting
  Jaguar server 6
  user-defined Jaguar server 6
stubs
  and proxy objects 2
Sybase Virtual University
  sample application 16
Index

T

termology
  component based applications  2
tutorial
  ActiveX  49
  C++  33
  Java  17
  security  65