SYBASE®

System Administration Guide

EAServer

Version 5.2

DOCUMENT ID: DC38034-01-0520-01

LAST REVISED: January 2005

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

Contents

About This Book

Subject

This book contains information about configuring and running EAServer.

Audience

This book is for anyone responsible for configuring the EAServer runtime environment, or for creating and deploying packages and components on EAServer.

How to use this book

Chapter 1, “Getting Started,” contains instructions for starting the preconfigured server, connecting to EAServer Manager, and changing the administration password.

Chapter 2, “Sybase Central Overview,” describes the graphical user interface tool, Sybase Central, and how to connect to the EAServer Manager plug-in, and the Certificates folder, which is used to manage the EAServer certificate database.

Chapter 3, “Creating and Configuring Servers,” contains information about configuring the EAServer runtime environment, including:

- Creating servers
- Setting server properties
- Setting HTTP properties

Chapter 4, “Database Access,” describes how to configure connection caches, XA resources, J2EE connectors, and transaction control.

Chapter 5, “Naming Services,” explains how to use EAServer naming services to locate objects—such as packages, components, and servers—anywhere on the network.

Chapter 6, “Clusters and Synchronization,” contains information about creating a cluster of servers, which provides high availability for EAServer services and components, and synchronizing repositories from a primary server within a cluster to other clustered servers, which keeps all of the servers within the cluster up to date.

Chapter 7, “Load Balancing, Failover, and Component Availability,” explains how to load balance between a cluster of servers and how to configure and implement component failover.

Chapter 8, “Setting up the Message Service,” describes how to set up the message service properties and add the message service parts that enable you to publish, send, and receive messages.

Chapter 9, “Importing and Exporting Application Components,” explains how to deploy packages and components, Web applications, J2EE applications, J2EE connectors, and application clients.

Chapter 10, “Using Repository Versioning” explains how to save numbered versions of EAServer’s repository, which stores configuration and implementation files for entities such as components, Web applications, JSPs, and Java servlets.

Chapter 11, “Runtime Monitoring,” describes how to use the File Viewer, Runtime Monitor, and the OTS Transaction Monitor to track EAServer’s performance and statistics.

Chapter 12, “Using jagtool and jagant,” contains information about the command-line tools that allow you to automate some of EAServer’s development and deployment tasks.

Chapter 13, “Using Systems Management” describes the EAServer systems management functionality—based on the JMX agent management framework—including how to use the Web console to view the status of services and agents, and how to monitor events. It also explains how the Systems Management framework allows SNMP clients to view information about underlying system status using standard SNMP protocols.

Appendix A, “Using EJB 1.0 JAR Support,” describes EAServer Manager’s support for EJB 1.0 JAR files.

Appendix B, “Repository Properties Reference,” describes the EAServer repository properties.

Related documents

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

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

The *EAServer Cookbook* contains tutorials and explains how to use the sample applications included with your EAServer software.

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

The *EAServer System Administration Guide* (this book) explains how to:

- Start the preconfigured Jaguar server and manage it with the EAServer Manager plug-in for Sybase Central™
- Create, configure, and start new application servers
- Define connection caches
- Create clusters of application servers to host load-balanced and highly available components and Web applications
- Monitor servers and application components
- Automate administration and monitoring tasks with command line tools or the Repository API

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

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

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

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

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

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

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

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

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

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

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

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

Conventions

The formatting conventions used in this manual are:

Formatting example	To indicate
commands and methods	When used in descriptive text, this font indicates keywords such as: • Command names used in descriptive text • C++ and Java method or class names used in descriptive text • Java package names used in descriptive text • Property names in the raw format, as when using jagtool to configure applications rather than EAServer Manager

Formatting example	To indicate
variable, *package*, or *component*	Italic font indicates: • Program variables, such as *myCounter* • Parts of input text that must be substituted, for example: *Server*.log • File names • Names of components, EAServer packages, and other entities that are registered in the EAServer naming service
File | Save	Menu names and menu items are displayed in plain text. The vertical bar shows you how to navigate menu selections. For example, File | Save indicates "select Save from the File menu."
`package 1`	Monospace font indicates: • Information that you enter in EAServer Manager, a command line, or as program text • Example program fragments • Example output fragments

Other sources of information

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

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

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

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Select Products from the navigation bar on the left.

3 Select a product name from the product list and click Go.

4 Select the Certification Report filter, specify a time frame, and click Go.

5 Click a Certification Report title to display the report.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Accessibility features

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

EAServer Manager supports working without a mouse. For more information, see "Keyboard navigation" on page 16.

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

1 Start Eclipse

2 Select Help | Help Contents

3 Enter Accessibility in the Search dialog box

4 Select Accessible user interfaces or Accessibility features for Eclipse

---

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

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

If you need help

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

CHAPTER 1

Getting Started

This chapter explains how to get started with EAServer using the preconfigured Jaguar server.

Starting the Jaguar server

Before you can develop EAServer applications, you must start the preconfigured Jaguar server.

❖ **Starting the preconfigured server in UNIX**

1. Add the location of the X-Windows *xterm* utility to your *path* variable. For example:

   ```
   set path = ($path /usr/local/SUNWmotif/bin/)
   ```

2. Edit the *$JAGUAR/bin/setenv.sh* shell script. Change the JAGUAR variable to the EAServer home directory.

3. You can run the Jaguar server in different modes, debug and normal, using different Java runtime versions and different Java VMs. Change to the *$JAGUAR/bin* directory and run the *serverstart.sh* script using this syntax:

   ```
   serverstart.sh [-jvmtype classic | client | server] \
   [-jdk13 | -jdk14] [-debug] [-workshop] [-xterm]
   ```

Option	Description
-jvmtype [classic | client | server]	Specify which Java virtual machine (VM) to use: • Classic – classic Java VM. • Client – HotSpot Client VM. • Server – HotSpot Server VM. If you run JDK 1.4, you can choose either of the HotSpot VMs; the default is the HotSpot Client VM. The classic VM is not supported with JDK 1.4. On Solaris, if you use JDK 1.3, you can run either of the HotSpot VMs.
-jdk13 | -jdk14	Specify the JDK version; the default is 1.3.
-debug	Run the server in debug mode.

-workshop	Solaris only – enables in-process debugging in the Workshop debugger. Define the location of the Workshop debugger by setting the WORKSHOP_DIR variable in the *user_setenv.sh* file; for example:
	`WORKSHOP_DIR=/OPT/SUNWspro6.2/`
	`export WORKSHOP_DIR`

-xterm	Run the server in a new window opened by xterm.

When you run *serverstart.sh* without any options, the Jaguar server runs with JDK 1.3 and the HotSpot Client VM.

❖ **Starting the preconfigured server in Windows**

- If the preconfigured server is not installed as a Windows service. Select Start | Programs | Sybase | EAServer 5.2. Choose one of these options:
  - Jaguar Server – starts the server using JDK 1.3 and the Java HotSpot Client VM.
  - Jaguar Server (JDK 1.4) – starts the server using JDK 1.4 and the Java HotSpot Client VM.
  - Jaguar Server (debug) – starts the debug server using JDK 1.3 and the classic Java VM.
  - Jaguar Server (debug JDK 1.4) – starts the debug server using JDK 1.4 and the HotSpot Client Java VM.

Debug servers Debug-mode servers allow you to remotely debug components from tools that support EAServer component debugging, such as PowerBuilder or PowerJ®. You cannot run the debug server unless you installed the debug libraries and binaries. The debug server cannot run as a Windows service.

To start a user-defined server, you must first create the server. For instructions, see "Creating or deleting a server" on page 19.

Using EAServer Manager

EAServer Manager runs within Sybase Central. Use EAServer Manager to configure EAServer and to define and deploy software components and packages.

EAServer must be running before Sybase Central can connect to it.

To use EAServer 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 additional security, you can establish a password for the jagadmin user. See "Administration password and OS authentication" on page 4.

For additional information on EAServer administrative privileges, see Chapter 13, "Security Configuration Tasks," in the *EAServer Security Administration and Programming Guide*.

Starting EAServer Manager

EAServer Manager is a plug-in for Sybase Central. You must start Sybase Central first, then connect to EAServer Manager from within it.

❖ **Starting Sybase Central in UNIX**

1 Edit the *$JAGUAR/bin/setenv.sh* shell script. Change the JAGUAR variable to the EAServer home directory.

2 Enter:

```
./jagmgr
```

❖ **Starting Sybase Central in Windows**

- Double-click the EAServer Manager icon in the EAServer program window, or select Start | Programs | Sybase | EAServer 5.2 | EAServer Manager.

Once the Sybase Central window appears, you can connect to EAServer Manager. EAServer Manager logs errors and other messages to the *$JAGUAR/bin/jagmgr.log* file.

❖ **Connecting to EAServer Manager**

1 Select Tools | Connect

2 On the Login window, enter `jagadmin` as the user name. Verify the host name or IP address, and the EAServer port number, and click Connect. The machine name or IP address and port number correspond to the EAServer host entry and IIOP port number defined in the listener.

Disconnecting from EAServer

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

❖ **Disconnecting from a server**

- Select Tools | Disconnect.

Administration password and OS authentication

Members of the Admin role have unlimited access to EAServer Manager. Initially, the jagadmin user is the only member of this role. For additional security, you can establish an administration password for the jagadmin user and enable operating system authentication.

To access and configure these properties:

1 From EAServer Manager, highlight the server you want to configure.

2 Select File | Properties.

3 Select the Security tab. The remainder of this section describes how to configure EAServer using the controls on this tab.

Administration You can establish an administrative password for the jagadmin user on each server. The jagadmin user can:

- Access EAServer Manager
- Set or reset the jagadmin password
- Enable and disable user authentication

To set the jagadmin password:

1 Select Set jagadmin Password.

2 In the Administrator Password dialog box, enter the old password, the new password twice, and click OK.

Administration password conventions and restrictions are the same as for user passwords for your platform.

Enabling OS authentication If selected, this option maps EAServer client users to operating system user names and passwords. You must supply a user name and password that is valid for the machine where EAServer is running. For example, for UNIX, you would use network information service (NIS) passwords, and for Windows, you would use your Windows domain password. Windows users can provide a domain name as part of their user name; for example, *domain_name**username*.

❖ **Enabling OS authentication on UNIX**

- Select the Enable OS Authentication option on the Security tab.

❖ **Enabling OS authentication on Windows 2000**

Users who run EAServer must belong to the Administrators Group on your Windows machine. Add users and groups who will start EAServer to the Administrators Group.

1 Select Start | Settings | Control Panel.

2 Double-click Administrative Tools.

3 Double-click Local Security Settings.

4 In the left pane, click Local Policies.

5 Select and open User Rights Assignment.

6 Double-click Act as Part of the Operating System.

7 Click Add in the new pop-up window to add the desired users. This provides the required privileges to EAServer to authenticate a user by querying the underlying operating system.

8 Log out, then log back in to your Windows 2000 system to enable authentication.

9 From EAServer Manager, select Enable OS Authentication on the Server Properties Security tab.

❖ **Enabling OS authentication on Windows XP**

Users who run EAServer must belong to the Administrators Group on your Windows machine. Add users and groups who will start EAServer to the Administrators Group:

1 Select Start | Settings | Control Panel.

2 If your Control Panel is in category view, double-click Performance and Maintenance.

3 Double-click Administrative Tools.

4 Double-click Local Security Policy.

5 Expand the Local Policies folder, then select User Rights Assignment.

6 Double-click Act as Part of the Operating System.

7 In the new dialog box, click Add User or Group to add users.

8 In the Select Users or Groups dialog box:

   a Click Object Types, and select Users.

   b Click Locations, and select the network domain.

   c Enter the user names.

   This provides the required privileges to EAServer to authenticate a user by querying the underlying operating system.

9 Log out, then log back in to your Windows XP system to enable authentication.

10 From EAServer Manager, select Enable OS Authentication on the Server Properties Security tab.

Note The password for the jagadmin account must be defined in EAServer Manager. Even if jagadmin is defined as an OS user name and OS authentication is enabled, the password defined in EAServer Manager is required to log in as jagadmin.

Enable User & Groups Validation If enabled, the user and group names are validated against their operating system user and group name before being added to any of the following folders:

- Authorized User
- Authorized Group
- Excluded User
- Excluded Group

To enable user and group validation, select the Enable User and Groups Validation option on the server's Security tab.

JAAS Configuration File To use Java authentication and authorization service, enter the name of a file that specifies:

- One or more authentication modules for an application
- The order in which to invoke the authentication modules
- Other parameters and options

For complete information about using JAAS, see Chapter 11, "Using the JAAS API," in the *EAServer Security Administration and Programming Guide*.

Security Identities Define a user name, password, and SSL session characteristics used by components or servlets that call other components. Identities are also used for inter-server authentication when propagating caller credentials in a call sequence that involves multiple servers. See "Intercomponent authentication for EJBs and servlets" in Chapter 2 of the *EAServer Security Administration and Programming Guide*.

Shutting down a server

1 Select Servers | *server_name*, where *server_name* is the server to which EAServer Manager is connected.

2 Select File | Shutdown *server_name*.

Unless you have changed the listener address for the server, you can remain logged in to EAServer 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.

Verifying your environment

If you have any problems running the Jaguar server or Sybase Central:

All platforms

- Check the server log file (*Jaguar.log*) in the *bin* subdirectory for error messages.
- Verify the server's HTTP, IIOP, and TDS port settings. Default listeners are defined for each protocol, but you may have changed them. See "Configuring listeners" on page 44 for more information.
  - TDS is the port used by EAServer to accept TDS requests. The default is 7878.
  - HTTP is the HTTP port used by EAServer to listen for HTTP requests. The default is 8080.
  - IIOP is the port used by EAServer to accept IIOP requests. The default is 9000.

Invalid listener configuration If you configure an invalid listener address (for example, by specifying an invalid host name), EAServer attempts to use an alternate address. When this occurs, the new address is recorded in the server log file.

UNIX

- Use the *setenv.sh* script to configure the environment.

  In the *$JAGUAR/bin* directory, the *setenv.sh* shell script is used to set up the environment. Make sure you have set the $JAGUAR variable.
- Verify that the THREADS_FLAG variable is either not set or set to a value that matches the setting in the *setenv.sh* script.

Windows

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.

- Verify that the EAServer installation was performed as user "Administrator."

- 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:\Sybase\EAServer\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:\Sybase\EAServer\include;c:\MSDEV\include
  ```

More information

For more information, see the EAServer Troubleshooting Guide at http://www.sybase.com/detail?id=1024509.

CHAPTER 2

Sybase Central Overview

This chapter describes Sybase Central, the graphical user interface tool that you use to manage EAServer applications and security.

Overview

Sybase Central is a common management framework for Sybase applications and database servers. EAServer provides EAServer Manager as a Sybase Central plug-in for use by developers and administrators.

EAServer Manager provides graphical administration facilities for EAServer, including support for development, deployment, and runtime monitoring of applications. You can manage SSL digital certificates on client and server machines. On machines with full EAServer installations, the Certificates folder controls the server's certificate database.

Start-up options

You must start Sybase Central first, then connect to EAServer Manager from within it. In an EAServer 5.0 or later installation, Sybase Central is located in *$SYBASE/sybcent41*. This location enables other Sybase products and multiple EAServer instances to use the same interface installation.

❖ **Starting Sybase Central on UNIX**

- To start Sybase Central, run either of these scripts:

```
$JAGUAR/bin/jagmgr

$SYBASE/sybcent41/bin/scjview
```

❖ **Starting Sybase Central on Windows**

- Select Start | Programs | Sybase | EAServer | EAServer Manager, or run either of these scripts from a command line:

```
%JAGUAR%\bin\jagmgr.bat

%SYBASE%\sybcent41\bin\scjview.bat
```

❖ **Connecting to EAServer Manager**

To use EAServer Manager, you must be the jagadmin user or have the Admin role.

1 Select Tools | Connect. A server must be running before Sybase Central can connect to it.

2 If you are connecting for the first time, use jagadmin as the user name and leave the password blank. Verify the EAServer host and port number, and click Connect. The host and port number correspond to the EAServer host entry and IIOP port number defined in the listener.

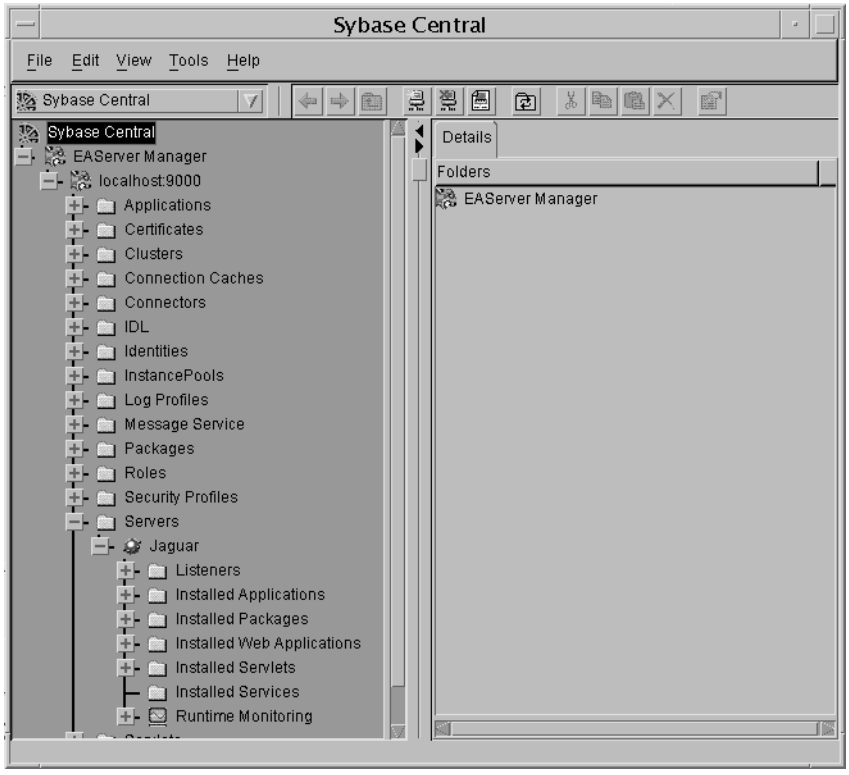
For security, you can establish a password for the jagadmin user—see “Administration password and OS authentication” on page 4.

EAServer and EAServer Manager must be the same version You can connect to EAServer version 5.2 from EAServer Manager version 5.2. While you can connect to earlier versions of EAServer from EAServer Manager 5.2, this configuration is not supported, and you may encounter problems. You cannot connect to EAServer 5.2 from earlier versions of EAServer Manager.

EAServer Manager

EAServer Manager runs within Sybase Central. Use EAServer Manager to configure EAServer and to define and deploy software components and packages. EAServer Manager is EAServer's graphical user interface tool that allows you to easily define the packages, components, and methods that EAServer clients use to run an application.

Figure 2-1: EAServer Manager



EAServer Manager also allows you to manage the server and user certificates that are required for SSL-protocol support—see "Certificates folder and the standalone Security Manager" on page 15.

Application developers can use EAServer Manager to view the method definitions for any installed component in EAServer Manager. You can view and edit method definitions graphically, or you can directly edit the interface definition language (IDL) datatype and interface definitions with EAServer Manager's IDL editor.

EAServer Manager also generates stub classes for use in Java and C++ client applications, and ActiveX type libraries for use in ActiveX client applications.

To simplify application deployment, EAServer Manager defines the following basic, middle-tier application units:

- Cluster – a set of servers that share configuration information and run the same set of components.
- Server – an EAServer runtime process with its own network addresses for client session connections and for HTTP (HTML) connections.
- Application – a group of packages and Web applications bundled into a single unit for easy deployment between servers.
- Package – a collection of components organized into cohesive, secure units that can be easily deployed on another EAServer installation.
- Component – contains the methods that execute business logic and access data sources.
- Web application – a unit of deployment for interrelated Web content, JavaServer Pages (JSPs), and Java servlets.
- Web component – a servlet or JSP installed in a Web application.

The Refresh menu option allows you to refresh components, packages, and servers, which lets you test and debug component implementation changes without restarting the server.

Logging errors

When you run EAServer Manager, logging and error messages are written to the *jagmgr.log* file, in EAServer's *bin* directory.

EAServer Manager enables you to remotely view server log files and to monitor statistics for component execution and network activity. For more information on runtime monitoring, see Chapter 11, "Runtime Monitoring."

Profile Manager

When you log in to EAServer Manager, you can use predefined login profiles, which speeds up the connection process. Define profiles using the Profile Manager, then select the profile from the drop-down list in the EAServer Manager login window and enter a password to connect to the port defined in the profile.

❖ **Defining a profile**

1 Open the Connection Profiles dialog box using either of these methods:

- From Sybase Central, select Tools | Connection Profiles.
- From the Login window, click Profile Manager.

2 Click New and define the following fields:

- Name – the name of the profile that displays in the Profiles drop-down list.
- Select whether to allow all users to access the profile.
- Select either New Profile or Copy Profile. If you select Copy Profile, also select the profile from which to copy the property values.

  Click OK.

3 In the Login dialog box, enter:

- User Name – the name of the user connecting to the port; for example, jagadmin.
- Password – 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.
- To use IIOPS for this profile's connections, select Secure Connection, then select either EAServer SSL or JSSE, and enter the PIN.

4 Click OK to add the profile.

To delete a profile, highlight the profile and click Delete.

Certificates folder and the standalone Security Manager

Use the Certificates folder on machines with full EAServer installations, or use the standalone Security Manager on client machines, to manage SSL certificates, the test CA, and public and private key pairs.

From the Certificates folder, you can manage the server and user certificates that are required for SSL-protocol support, which allows you to:

- Install server certificates – required to establish secure IIOP and HTTP ports.
- Install Certificate Administrator (CA) certificates – also called **signing certificates**, are attached to client and server certificates to validate the origin of the certificate.
- Issue certificates for testing purposes – you can create new client and server certificates to test your applications.

For detailed information on EAServer administrative privileges and the Certificates folder, see the *EAServer Security Administration and Programming Guide*.

Keyboard navigation

EAServer Manager and Security Manager both allow you to work without a pointing device. You can navigate with the keyboard as follows:

- **To activate menus** Press the Alt key plus the underlined menu hotkey; for example, Alt+T displays the Tools menu. To navigate within a menu, use the arrow keys to move between items, and press Enter to activate the selected item. You can also press Alt plus the underlined hotkey to activate items within the menu.
- **To move the focus between main window components** Press the Tab key to move the focus between parts of the main window. The tab order is:

  a The active component pull-down, in the upper-left corner below the menu bar.

  b The tree view, below the active component pull-down.

  c Details view header, which is the single details tab over the right pane, where details show for the item you have highlighted in the tree view.

  d Details view data, the right window pane where Sybase Central shows detail items for the item you have highlighted in the tree view.

  Press Shift+Tab to move the focus to the previous component in the list.

- **To navigate in the tree view and details view** The tree view, in the left pane of Sybase Central, allows you to navigate through the entities and interfaces that can be configured in EAServer. The details view shows detail items for the item that is highlighted in the tree view. To navigate through items in these view, first move the focus to the tree view or Detail view, then use these keys:
  - Up arrow and down arrow – move between items in the tree. You can also type the first letter of the item, or the first two letters if the first letter is not unique. If items duplicate more than two initial letters, use the up and down arrow to navigate past the first match.
  - Enter – collapse or expand items in the tree view.
  - Alt+Enter – show properties for the selected item if the item has a Properties menu item. For example, components and servers.
  - Alt+*hotkey* – activate menu items for the selected item.
- **To navigate in dialog boxes** To navigate between controls in dialog boxes, press the Tab key to move the focus to the desired control. In tabbed dialogs, press Ctrl+Tab to move between tabs. You can activate or move the focus to a control by pressing Alt plus the control's underlined hotkey. To activate buttons, place the focus on the button and press the spacebar. You can display online help for the displayed dialog by pressing F1 or by placing the focus on the help button and pressing the spacebar. You can cancel the dialog by pressing Escape or navigating to the Cancel button and pressing the spacebar.

  Note In most cases, pressing Enter activates the OK button. The exception is when the focus is on another button that is not a Next, Previous, Finish, or Cancel button. With the focus on these buttons, pressing Enter may activate the button with the focus. Any button can be activated by navigating the focus to it and pressing the spacebar.

- **Navigating in the help viewer** Table 2-1 describes keyboard navigation in the help viewer.

Table 2-1: Keyboard navigation in the help viewer

To do this	Use these keys
Navigate the focus through the toolbar buttons, table-of-contents pane, and search pane	Ctrl+Tab to move the focus forwards and Ctrl+Shift+Tab to move backwards. On most systems, toolbar buttons indicate that they have the focus by showing a dotted line around the button label. You can also press Ctrl+F1 to show tooltip text for the button that has the focus. When the table-of-contents pane has the focus, there is a dotted line around the book icon in the tab. When the search pane has the focus, it displays over the table-of-contents pane with a blinking cursor in the search text input field.
Navigate between topics displayed in the table-of-contents pane	Place the focus in the table-of-contents pane as described above. The text pane displays the text associated with the highlighted topic in the table of contents. Use the up and down arrow keys to display the previous and next topics, respectively. You can use Page Up, Page Down, Home, and End to scroll quickly through the list of topics. Topics with a + sign can be expanded to display subtopics. To navigate to subtopics, highlight the parent topic, press Enter, then navigate with the down and up arrow keys.
Search for text	Place the focus in the search pane as described above. Type the text to search for, then press Enter. When the results appear, the first topic is displayed. To navigate through the topics, press Tab once to move the focus to the list of results, then use the Up Arrow, Down Arrow, Page Down, Page Up, Home, and End keys to scroll through the results.
Place the focus in the text (right) pane	Place the focus in the table-of-contents pane or search pane, then press F6. Press F6 again to move the focus back to the left pane. Alternatively, place the focus in the text entry field on the search pane, then press Tab twice.
Navigate in the text pane	Place the focus in the text pane as described above. To scroll, use the Page Up, Page Down, Down Arrow, Up Arrow, Home, and End keys. To move the focus to hyperlinks, use Ctrl+T to move the focus to the next hyperlink, and Ctrl+Shift+T to move the focus to the previous hyperlink. To follow the hyperlink that has the focus, press Ctrl+spacebar.
Activate a toolbar button (Back, Forward, Print, Page setup)	Place the focus on the button, press the spacebar.
Move the splitter bar.	Place the focus in the table-of-contents pane, search pane, or text pane. Press F8 to place the focus on the splitter bar. Use the Left Arrow and Right Arrow keys to move the splitter bar.

CHAPTER 3

Creating and Configuring Servers

This chapter describes basic configuration tasks that you can perform to customize your installation, such as creating new servers, changing server properties, and customizing your environment.

The EAServer runtime environment is preconfigured; with minimum setup, you can have a fully functioning transaction server. Although the default settings are usually sufficient, EAServer allows you to customize your server environment as necessary.

You can perform all configuration tasks using the EAServer Manager.

Creating or deleting a server

❖ **Creating a server**

1 Double-click the EAServer Manager icon.

2 Highlight the Servers folder.

3 Select File | New Server (or File | New Server Wizard).

4 Enter a name for the new server. Server names must be one word, and can be up to 255 characters long. Click Create New Server.

5 Complete the server configuration. For information, see “Configuring servers” on page 20.

6 For each new server you create, add an HTTP, TDS, and IIOP listener. See “Preconfigured listeners” on page 44 for more information.

7 Start the server. See “Starting the server” on page 49.

The server logs error and other messages to the *bin\server_name.log* file, where *server_name* is the name of the server.

❖ **Deleting an existing server**

1 Double-click the EAServer Manager icon.

2 Double-click the Servers folder.

3 Right-click the server you want to delete and click Delete.

4 Select File | Delete Server.

Note You cannot delete the server to which EAServer Manager is connected. At least one server must be defined in your EAServer installation.

Configuring servers

EAServer provides property tabs and three wizards that step you through the process of configuring and tuning a server. You can use property tabs, the wizards, or both, to configure and tune a server. The wizards step you through the configuration process, and the property tabs allow you to select which properties you want to configure in the order you choose. The wizards and property tabs modify the same information. The Advanced tab displays all server settings.

❖ **Configuring a server using the Server Configuration wizard**

The Server Configuration wizard steps you through the basic configuration required to establish a server.

1 From within EAServer Manager, display the list of installed servers by expanding the Servers folder.

2 Highlight the server you want to configure.

3 Select File | Server Configuration Wizard.

❖ **Tuning the server using the Performance Tuning wizard**

The Performance Tuning wizard fine-tunes your server for optimum performance.

1 From within EAServer Manager, display the list of installed servers by expanding the Servers folder.

2 Highlight the server you want to configure.

3 Select File | Performance Tuning Wizard.

❖ **Configuring debug settings using the Server Debug Settings wizard**

The Server Debug Settings wizard defines additional settings for debugging server and performance problems.

1 From within EAServer Manager, display the list of installed servers by expanding the Servers folder.

2 Highlight the server you want to configure.

3 Select File | Server Debug Settings Wizard.

❖ **Configuring or modifying server properties**

Select each tab as required to define various aspects of server behavior.

1 From within EAServer Manager, display the list of installed servers by expanding the Servers folder.

2 Highlight the server you want to configure.

3 Select File | Properties. The Server Properties dialog box displays, which contains these tabs:

- General – define general individual server properties.
- Java VM – control the execution of Java in the server.
- HTTP Config – determine browser accessibility.
- Transactions – determine the transaction coordinator for components that participate in EAServer transactions.
- Security – see “Administration password and OS authentication” on page 4 for security options accessible from this tab.
- Resources – define the maximum number of client connections.
- Log/Trace – set logging and trace options.

- Handlers – if hosting an Open Server application, allows you to specify Open Server event handler function names.
- Naming Service – set the server's naming service options. See Chapter 5, "Naming Services" for additional information.
- Servlets – disable servlet execution in EAServer and configure additional properties to control the execution of servlets.
- PowerDynamo – enable hosting of PowerDynamo Web sites in EAServer.
- Hot Standby – enable hot standby and define the master and backup servers.
- JAXP Support – configure Java API for XML parsing support.
- Java Classes – specify classes to include in the server's runtime JAR file.
- Java Debug – specify the port number for remote debugger connections.
- Static Page Caching – enable caching for static pages.
- HTTP Custom Response Header – define custom headers for HTTP responses.
- HTTP Directory Browsing – enable HTTP browsing for Web application and non-Web application directories.
- Components – configure component execution.
- Advanced – edit server property settings in their raw format, that is, as they are stored in the configuration repository.

If you modify any property, click OK in the Server Properties dialog box to save your changes, or click Cancel to discard the changes.

When you modify server properties, you must refresh the server for the changes to take effect. To refresh the server, highlight the server icon and select File | Refresh.

General

Table 3-1 describes the general properties that you can configure for individual servers.

Table 3-1: Server general properties

Property	Description	Comments
Description	Enter a description of the server, up to 255 characters in length.	
Codeset	Specify the character set used by the server.	By default, the server uses utf8. For the list of supported values, list the subdirectories of the *charsets* directory. Each subdirectory matches the name of a supported character set.
Classpath	Displays the contents of the CLASSPATH environment variable for the server that you are connected to. This setting specifies the directories from which Java class files can be loaded. It is defined by the start-up script when you start the server. CLASSPATH does not display for servers that you are not connected to.	This setting is read-only and helpful for debugging various errors. To change the value, you must reset the environment variable and restart the server.
System Environment Variables	Displays the values of the system environment variables. These can be defined in the script that you use to start the servers, in system files, such as *.login*, or on the Advanced tab in the Windows System Properties dialog box.	This property is read-only. To change a value, you must reset the environment variable and restart the server.

Java VM

The settings on this tab control a variety of parameters that affect the execution of Java in the server. Table 3-2 lists the settings. If you change these settings, you must restart the server before the changes take effect.

Note To change the JDK version and VM type used in the server, specify them with the command-line arguments described in “Starting the server” on page 49.

Table 3-2: Java VM properties

Property	Description	Default value
Boot Classpath	Specifies the directories and JAR files in the boot class path search list for the Java virtual machine. Classes loaded from these locations can override the core Java runtime classes. In most cases, the value should match the class path setting. The syntax is the same as for the class path setting.	${BOOTCLASSPATH}, which is replaced by the value of the BOOTCLASSPATH environment variable at runtime.

Property	Description	Default value
Boot Library Path	Specifies the directory search path to load native libraries used by Java classes. The syntax is the same as for the platform PATH environment variable setting.	${BOOTLIBRARYPATH}, which is replaced by the value of the BOOTLIBRARYPATH variable at runtime.
Classpath	Specifies the class path for the Java virtual machine, which is the list of directories and JAR files that are searched to load classes. These locations are searched after the boot class path locations and cannot override the core Java runtime classes. The list consists of directories and the full path to JAR files. On Windows platforms, use a semicolon (;) to separate entries. On UNIX platforms, use a colon (:).	${CLASSPATH}, which is replaced by the value of the CLASSPATH variable at runtime.
Use Jaguar Class Loader Version 2	If selected, EAServer uses version 2 of the custom class loader. For more information on class loader versions, see Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*.	Not selected, which indicates that EAServer uses the version 1.0 class loader.
Display Options at Startup	If this option is selected, the server logs all the Java VM options when starting.	Not selected.
Disable JIT	If this option is selected, the Java just-in-time (JIT) compiler is disabled. The JIT compiler converts Java bytecode to native machine code, which can execute significantly faster.	Not selected.
System Class Loader Tracing	Enables verbose logging in the system class loader. The log output includes the name and source location for each class loaded.	Not selected.
Custom Class Loader Tracing	Enables verbose logging in the custom class loader. In the default configuration, component and Web application classes are loaded by the custom loader to allow refresh of your application code without restarting the server.	Not selected.
Verbose Garbage Collection	Enables logging of Java garbage collector activity.	Not selected.
Verbose JNI	Enables verbose logging of Java Native Interface (JNI) method linking and execution.	Not selected.
System Variables	Read-only list of system variables that are set inside the Java VM.	

Default paths for clusters If you use a cluster, Sybase recommends that you use the default values of class path, boot class path, and boot library path. If you enter paths in these properties, they must be valid on all machines in the cluster.

HTTP Config

Clients can access EAServer and retrieve HTML pages using a Web browser. You can customize certain aspects of your server's HTTP behavior by modifying the HTTP configuration properties listed in Table 3-3.

Table 3-3: HTTP properties

Property	Description	Default value	Comments/example
Domain Name	Domain name in *.company.xxx* format. Set this only if you are configuring the redirection URL for use with a Web proxy.	N/A	See "Configuring redirection addresses when using a proxy server" on page 27.
Proxy HTTP Port	When a Domain Name is specified, the HTTP port used in redirection URLs.	80	See "Configuring redirection addresses when using a proxy server" on page 27.
Proxy HTTPS Port	When a Domain Name is specified, the HTTPS port used in redirection URLs.	443	See "Configuring redirection addresses when using a proxy server" on page 27.
Proxy Protocol	When a Domain Name is specified, the protocol for redirection URLs.	The protocol of the original request	See "Configuring redirection addresses when using a proxy server" on page 27.
Document Root	The path to the directory where documents are served.	*$JAGUAR/html* (UNIX) *%JAGUAR%\html* (Windows)	*/work/WWW/* *C:\work\WWW*
Default Web Application	The default Web application.	None	The Web application must be installed in the server. When specified, clients' requests are redirected to the Web application context. For example, if the default Web application is MyWebApp, requests for `http://myhost:8080` are redirected to `http://myhost/MyWebApp`, and the welcome file is used.

Property	Description	Default value	Comments/example
Maximum Threads	The maximum number of threads allocated for HTTP requests. **Warning!** If you increase this value, you must also increase the maximum number of threads on the Resources tab—see "Resources" on page 30.	25	The maximum thread setting allows you to balance memory resources. A maximum value set too high needlessly uses memory resources. Monitor the total number of hits listed in the *httpstat.dat* file for indications of a heavily loaded server. Adjust the maximum thread setting as necessary.
Send "Server" Header in HTTP Response	If selected, EAServer adds the "Server" response header field to each HTTP response.	Disabled	This optional HTTP response header field contains a description of the server software.
Keep Statistics	Select to log statistics.	Disabled	
Statistics File Name	If you select to keep statistics, specify the log file name.	*Jaguarhttpstat.dat*	*/work/logs/Jaguarhttpstat.dat* (UNIX) *C:\work\logs\Jaguarhttpstat.dat* (Windows)
Frequency (Seconds)	If you select to keep statistics, specify how often to log them.	36000 seconds (10 hours)	
Log Type	You can select All Logs or one of these log types: • Request log • Error log	All Logs	If you select All Logs, the directory, file size, and truncate options apply to all the log files.
Enable Logging	Select to enable logging.	Enabled	
Log Directory	The directory where the log files are stored.		See "HTTP logging and statistics" on page 29 for information about the log files.
Log File Size	The size, in bytes, to which the log file grows before it is truncated.	Unlimited. If you do not enter a value, log size is unlimited.	
Truncate Log on Startup	When this flag is set, the log truncates every time the server is restarted.	The default is to not truncate on start-up.	If the server crashes and this flag is set, you will lose the log file and the information it contains.

Property	Description	Default value	Comments/example
Extended Log File Format	If enabled, EAServer writes to the request log using the extended log file format (ELFF), instead of the common log format.	Disabled. By default, common log format is used for the request log.	
ELFF Items	When ELFF is enabled, specifies what items to include in each record.	See comments.	For the syntax and default values, see com.sybase.jaguar.server.http.elffitems on page 527.

Configuring redirection addresses when using a proxy server

You may need to configure redirection addresses if clients connect to your server through a proxy or firewall, or if you want the domain name set in cookies that are returned to the client.

Some HTTP requests may cause EAServer to redirect the client to a different page. For example, when a client requests a page that has access constraints, the user may be redirected to the login page. The HTTP redirect response code contains a fully qualified HTTP or HTTPS URL, including the protocol, host, and port. When clients connect through a proxy server, the redirection URL must be modified to specify the proxy host address rather than the EAServer host address. Some proxies do not filter any packets in the base protocol. A hardware SSL engine is one such example. When using these proxies, you must configure the properties below to ensure that the proxy address and protocol appear in the redirection URL, rather than the address used for the connection from the proxy to EAServer:

Using the Web server redirector When you are using the EAServer Web server redirector, you need not set these properties. The redirector automatically modifies the HTTP Location response header value, provided the host and port in the value match an EAServer that the redirector is configured to service. If you do set the proxy properties, the redirector does not modify the HTTP Location response header value.

- **Domain Name** If this property is set, EAServer composes redirection URLs using the specified host or domain name rather than the host name set in the EAServer listener. The domain name is also set for cookies that are returned to the client. This allows sharing of cookie information across multiple servers in your domain. If the domain name is not set for the cookie, the information is available only to applications running on the server where it was created.

There are two ways to set the property:

- If you specify a domain name beginning with a dot, the redirection host is "www" at the specified domain. For example, if you enter `.foo.com` and EAServer runs on host abc, the redirection host is www.foo.com, rather than abc.foo.com. Use this format if your proxy or firewall server runs with host name "www."
- If you specify a value that does not begin with a dot, the redirection host is the supplied value, with no domain name appended. In this case, enter either an IP address, or the entire address, including the domain name. For example, `search.foo.com` or `10.22.241.101`. Use the full name if clients do not connect through a proxy, or connect through a proxy with a host name other than "www."

When a domain name is specified, the address used in the location header for redirect response packets is composed as follows:

protocol`://`*address*`:`*port*

where:

- *protocol* is the value of the Proxy Protocol setting, or if not set, the protocol of the original request from the proxy to EAServer, which may not match the protocol used by the base client's connection to the proxy.
- *address* is the value of the Domain Name setting if the setting does not begin with a dot; otherwise, *address* is the Domain Name setting with "www" inserted at the beginning.
- *port*, for HTTP URLs, is the value of the Proxy HTTP Port setting, or 80 if not set. For HTTPS URLs, *port* is the value of the Proxy HTTPS Port setting, or 443 if not set.

To override the Domain Name for Web applications, you can set the com.sybase.jaguar.webapplication.httpdomain.override property to true. See Web application properties on page 575 for a description of this property.

- **Proxy Protocol** When a domain name is specified, the protocol for redirection URLs. Specify HTTP or HTTPS. If not set, the protocol matches that of the request sent from the proxy to EAServer. This may not match the original protocol. For example, the client may use HTTPS to connect to the proxy, which uses HTTP to connect to EAServer. In this case, you must set the Proxy Protocol to HTTPS. Otherwise, the client Web browser may refuse to connect to the redirected URL because of the switch from HTTPS to HTTP.

- **Proxy HTTP Port** When a domain name is specified, this setting specifies the port to embed in HTTP redirection URLs. The default is 80. When a domain name is not specified, this setting is ignored.
- **Proxy HTTPS Port** When a domain name is specified, this setting specifies the port to embed in HTTPS redirection URLs. The default is 443. When a domain name is not specified, this setting is ignored.

HTTP logging and statistics

EAServer maintains three HTTP log files and a statistics data file that allow you to monitor HTTP events. The file names are prepended with the server name. For example, if you create a server named Test_server, error messages for that server are directed to the *Test_serverhttperror.log* file. By default, the log files are located in the EAServer *bin* subdirectory (or *devbin* if you are running the debug server version).

- *<server_name>httprequest.log* – records HTTP request log information about each HTTP request.
- *<server_name>httperror.log* – logs HTTP errors, such as a request for an HTML file that does not exist.
- *<server_name>httpservlet.log* – logs HTTP servlet requests.
- *<server_name>httpstat.dat* – records HTTP statistics.

For information on viewing these files, see “Using the File Viewer” on page 205.

Transactions

This section describes the transaction coordinator models that are available. All components installed in one instance of EAServer share the same transaction coordinator.

EAServer transaction coordinator models are:

- **Java Transaction Service (JTS)** For UNIX or Windows users, this option complies with the JTS and the Object Transaction Service (OTS) and X/Open Architecture (XA) standards. The JTS transaction coordinator integrates the functionality of the shared connection, OTS/XA, and JTS/JTA transaction modes. The JTS transaction coordinator uses two-phase commit to coordinate transactions among multiple databases.

- **Microsoft Distributed Transaction Coordinator (DTC)** DTC uses two-phase commit to coordinate transactions among multiple databases. DTC is available on Windows platforms as part of Microsoft SQL Server 6.5 or later.

To set the transaction coordinator for your server, select the transaction model from the server's Transactions tab in the Server Properties dialog box.

For detailed information about components and transactions, see Chapter 2, "Understanding Transactions and Component Lifecycles," in the *EAServer Programmer's Guide*.

Resources

The Resources tab allows you to limit the number of concurrent client sessions and contains configurable properties used by Open Server applications. Table 3-4 describes the server resource properties.

Table 3-4: Server resource properties

Property	Description	Default
Maximum Number Client Sessions	The maximum number of concurrent client sessions supported by EAServer. This does not include HTTP sessions, which are controlled by the maximum thread property described in "HTTP Config" on page 25. Modify this variable as needed to balance system resources versus session availability. **Warning!** If you increase this value, you must also increase the maximum number of threads.	30
Message Pool Size The Open Server property SRV_S_MSGPOOL	The number of messages available to an Open Server application at runtime.	These properties are for Open Server applications. See your Open Server documentation for additional information.
Message Queue Size The Open Server property SRV_S_NUMMSGQUEUES	The number of message queues available to an Open Server application.	
Network Buffer Size The Open Server property SRV_S_NETBUFSIZE	The maximum size of the network I/O buffer to be used for TDS and Open Server listeners.	

Property	Description	Default
Maximum Number Threads	The maximum number of connection threads, including HTTP and IIOP connections and message service threads. Set this value equal to, or greater than, the sum of the maximum number of HTTP connections, the maximum number of client sessions, and the number of threads in the message service thread pools. See "HTTP Config" on page 25 for information about the HTTP connections value. See "Thread pools" on page 174 for information about configuring message service thread pools.	50
Thread Stack Size (available on UNIX platforms only)	The stack size for server threads, specified in bytes as a decimal number.	See "Configuring server stack size" on page 68 for information on setting this property.
Memory Usage/ Alarm Level	The percentage of system memory that can be used before the server begins blocking external requests.	70
Memory Usage/ Critical Level	The percentage of system memory that can be used before the server blocks external requests and begins cancelling in-process requests to bring usage below the specified critical threshold. For more information on memory usage settings, see Chapter 9, "Using the Performance Monitor," in the *EAServer Performance and Tuning Guide*.	90

Log/Trace

Tracing provides information about activities carried out by your application. Trace output is sent to the server's log file. To establish the level of detail for logging and tracing, select the Log/Trace tab. Table 3-5 describes the logging and trace properties.

For information on viewing the log file, see "Using the File Viewer" on page 205.

Table 3-5: Log/Trace properties

Property	Description
Logging Profile (Debug Server)	The log profile for the debug server, which specifies how and where errors and messages are logged. See "Configuring log profiles" on page 56 for more information.
Logging Profile (Production Server)	The log profile for the production server.
Trace Attentions	If set, traces attentions received or acknowledged by EAServer.

Property	Description
Trace Network Driver APIs	If set, traces Net-Lib driver requests.
Trace Network Driver Requests	If set, traces network layer protocol requests.
Trace Protocol Data	If set, traces TDS packet content (the actual TDS traffic between a client and EAServer) in hexadecimal and ASCII format.
Protocol Headers	If set, traces TDS protocol packet header information, such as packet type and length.
Trace Servlets	If set, traces the execution of EAServer's servlet execution engine.

Handlers

This tab allows you to specify Open Server event handlers when you have configured EAServer to support Open Server client connections. This feature allows you to run legacy Sybase Open Server applications in EAServer. For more information, see Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*. These handlers are called only for events that are generated by clients that connect to a listener that is configured to accept Open Server requests.

❖ **Specifying an event handler**

1 Select the Handlers tab.

2 Enter the DLL or shared library name and the function name of the specific event handler being called, separated by a colon.

The following example illustrates an entry for a connect event handler:

Platform	Entry
Solaris, AIX, Digital UNIX, and LINUX	*libsamp.so*:*debug_connect*
HP-UX	*libsamp.sl*:*debug_connect*
Windows	*libsamp.dll*:*debug_connect*

where *libsamp* is the DLL or shared library name and *debug_connect* is the function called whenever a connect event handler is called.

Table 3-6 summarizes the types of event handlers that you can install. For information on coding event handlers, see Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*.

Table 3-6: Individual server event handlers

Event handler	Called
Connect	Each time a client connects to EAServer.
Disconnect	When the client disconnects from EAServer.
Error	When a server processing error occurs.
CS Error	When a CS-Library error occurs
CT Error	When a Client-Library error occurs
Open Server Error	When a Open Server error occurs.
Initialization	Before starting a server.
Start	When a request to start the server is made.
Stop	When a request to stop the server is made.
Language	When a client sends a language request, such as a SQL statement.
RPC	When a client issues a remote procedure call.
Attention	When an attention has been received. An attention is an immediate event; EAServer services the attention as soon as it occurs, rather than adding it to the client's event queue.
Cursor	When a client sends a cursor request.
Dynamic	When a client sends a dynamic SQL request.
Message	When the client sends a message.
Option	When a client sends an option command.
Bulk	When a client issues a bulk copy request.

Naming Service

Select the Naming Service tab in the Server Properties dialog box to set the server's naming service options. You can use this property sheet to configure a server to be a name server, or to point to another server as its name server.

Note You can also set the bindpassword server property to enable password protection for name binding on a name server. For more information, see "Name binding password security" on page 119.

For general information about naming services, see Chapter 5, "Naming Services."

Initial Context – enter the server's default name context. The name server binds any object implementations on the server to the server's initial name context.

If you use an EAServer as a name server, the name context can be a compound name with each organization level separated with a forward slash ("/"); for example, */us/sybase/finance*.

If you use an external LDAP server to provide persistent storage, the initial context must match the schema used by the LDAP server. For example, *c=us,o=sybase,ou=finance*.

Naming server options

Use these options to specify whether the EAServer instance is also a name server, or whether it uses another EAServer instance as its name server.

- Click Enable as a Name Server to configure the server as a name server. If you select this option, you can then also set the Name Server Strategy options described below.
- If EAServer uses another EAServer instance as its name server, unselect Enable as a Name Server. Enter the URL for the EAServer instance acting as the name server; for example, `iiop://myhost:9000`.

Naming server strategy

If you enabled the EAServer instance to be a name server, indicate whether the server provides transient or persistent object name storage. By itself, an EAServer name server provides transient storage. However, you can add persistent storage capabilities to EAServer by using an external naming service, such as an LDAP name server.

If you enable persistent storage, enter the following information:

- The URL of the LDAP name server
- A manager DN (distinguished name) for the LDAP server
- The manager DN password

The manager DN provides exclusive access to all objects in the LDAP server database in order to bind and update the objects on the name server. The manager DN and its password are part of the LDAP server configuration properties, set by the server administrator. See your LDAP server documentation for complete information.

Servlets

On the Servlet tab in the Server Properties window, you can disable servlet execution in EAServer and configure additional properties to control the execution of servlets.

See Chapter 22, "Creating Java Servlets," in the *EAServer Programmer's Guide* for complete information about developing and configuring servlets.

PowerDynamo

This section discusses how to configure EAServer to host your PowerDynamo Web sites and provide access to those sites from a browser. Access to PowerDynamo Web sites is disabled by default.

❖ **Hosting PowerDynamo Web sites in EAServer**

1 Install PowerDynamo version 3.6.1 on your Windows or UNIX machine where EAServer is installed.

2 Update your EAServer's environment to include the PowerDynamo DLL and class files. Depending on your platform, perform the following:

On UNIX, make the following modifications to your *$JAGUAR/bin/setenv.sh* file, then source the file:

- Define a PDYNAMO environment variable and set it to the root of your PowerDynamo installation. This environment variable is required on UNIX machines.
- Define a JAGUARCLASSES environment variable and set it to your client JAR file, *$JAGUAR/client/easclient.jar*.
- Add *$PDYNAMO/lib* to your LD_LIBRARY_PATH.
- Add *$PDYNAMO/class03* to your CLASSPATH.

On Windows, depending on the virtual machine you are using, modify the *%JAGUAR%\bin\user_setenv.bat* file, or create this file if it does not exist. Make the following edits:

- Add *%PDYNAMO%\win32* to your PATH.
- Add *%PDYNAMO%\class03* to your CLASSPATH, where %PDYNAMO% is the root of your PowerDynamo installation. You do not need to define a PDYNAMO environment variable.

3 Start EAServer and connect to it from EAServer Manager. To enable PowerDynamo support:

   1 From EAServer Manager, open the Servers folder.

   2 Highlight the server you want to configure.

   3 Select File | Properties.

   4 Select the PowerDynamo tab.

   5 Click Enable PowerDynamo Execution.

   6 If you have both a PowerDynamo mapping and a servlet alias with the same URL, select Dynamo or Servlets from the Priority of EAServer HTTP Services list. This determines whether the PowerDynamo Web site or servlet is served to the client.

   7 Click OK.

4 Configure your machine so that your Web sites can connect to and retrieve information from databases that they use. For example, on Windows, if you load a Web site that accesses data from a SQL Anywhere database, you must include the SQL Anywhere DLLs in your PATH and set up the ODBC data source properly. See your database software instructions and the *PowerDynamo User's Guide* for detailed information.

Warning! If you have a Netscape Web server installed on your machine, PowerDynamo loads a Netscape version of *ns-httpd30.dll* instead of an EAServer version of the same DLL. Rename (but do not delete) Netscape's version of this DLL so that PowerDynamo loads the EAServer version instead.

You can now access a PowerDynamo Web site by entering into your browser, the EAServer HTTP address followed by a PowerDynamo Web site. For example:

`http://`***`EAServer_server_host`***`:8080/`***`mapped_url_name`***`/`***`file_name`***

This example connects your browser to:

- EAServer's HTTP port 8080 on the host machine identified by *EAServer_server_host*.
- The *mapped_url_name* is the mapping you supply for a PowerDynamo Web site in the PowerDynamo | Utilities | Configuration | Mappings folder.
- *file_name* is the file you are accessing from the mapped PowerDynamo Web site.

Hot Standby

If you have two EAServer installations, you can enable hot standby, which allows you to designate one of the servers as a backup server that accepts client connection requests in case the master server fails. The master server processes client requests. The backup server starts in "Admin" mode and does not accept client requests. If the master server fails or is unreachable, the backup server sets itself to "Ready" mode and accepts client requests. Once the master server is up and accepting requests, the backup server enters "Admin" mode, refusing connections from clients.

See Chapter 7, "Load Balancing, Failover, and Component Availability" for information about component failover.

❖ **Enabling hot standby in EAServer**

1 You must first enable the two hot standby servers as name servers. Select the Naming Service tab and click Enable as a Name Server to configure EAServer as a name server.

2 Select the Hot Standby tab and click the Enable Hot Standby check box.

3 Enter the Master Server URL using the format `iiop://`***`hostname`***`:`***`port`***. For example, `iiop://EAServer_master:9095`.

4 Enter the Backup Server URL using the format `iiop://`***`hostname`***`:`***`port`***. For example, `iiop://EAServer_backup:10000`.

The master and backup servers must be valid IIOP or IIOPS URLs. You can have only one master and one backup server defined and one of them, but not both, must be defined on the local server.

5 Synchronize the servers using the master as the primary server—see "Synchronization" on page 131.

You can verify the settings of hot standby by checking these properties on the Advanced tab:

- com.sybase.jaguar.server.CosNaming.nameserver must be set to true for both the master and backup servers.
- com.sybase.jaguar.server.hotstandby is set to true if hot standby is enabled.
- com.sybase.jaguar.server.hotstandby.master is the URL of the hot standby master server.
- com.sybase.jaguar.server.hotstandby.backup is the URL of the hot standby backup server.

Licensing requirements Hot standby requires two server deployment licenses, or a separately priced hot-standby license. Contact your Sybase sales representative for hot-standby licensing and pricing details.

JAXP Support

The JAXP Support tab allows you to configure the default Java XML parsers for components and Web applications running on the server. See Chapter 36, "Configuring Java XML Parser Support," in the *EAServer Programmer's Guide* for more information.

Java Classes

The Java Classes tab allows you to configure the set of classes to be custom-loaded at the server level. For more information, see "Custom class lists for packages, applications, or servers" in Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*

Java Debug

This tab configures the server's remote debugging interface with the settings listed in Table 3-7. Use the interface that your remote debugging tools support.

Table 3-7: Java Debug Settings

Setting	Specifies
Use JPDA	If selected, the server supports the JPDA (Java Platform Debug Architecture) interface using the specified port and the JPDA dt_socket transport type.
Port	The port number for JPDA debugger connections.
Use Agent	If selected, the server supports Java debugging using the sun.tools.debug interface.

After changing these settings, restart the server in debug mode for the change to take effect. For information on debugging Java components, see the *EAServer Programmer's Guide*.

Static Page Caching

You can configure EAServer to cache HTML and other static pages, which improves the speed at which the page contents are served. When a client requests an HTML page, EAServer checks the cache for a suitable entry. If the cache entry exists, the cached page is sent to the client. If the cache entry is not found, EAServer loads the page from disk, sends it to the client, then saves the page contents in the cache. Table 3-8 describes the static page caching options.

To cache dynamic content such as JSP or servlet responses, use one of the other caching options described in Chapter 5, "Web Application Tuning," in the *EAServer Performance and Tuning Guide*.

Table 3-8: Static page caching properties

Property	Description	Default value	Comments/example
Enable Page Caching	Select to enable page caching.	Enabled	
Enable Server Log Debug Messages	Select to log cache-related messages in the server log file, *server_name.log*.	Disabled	

Property	Description	Default value	Comments/example
Cache Size	The maximum size of the cache, specified in bytes, kilobytes, or megabytes, which you set using an integer plus “B”, “K”, or “M”.	10M	You can specify the size using uppercase or lowercase letters; for example, to set the cache size to 20 megabytes, you can enter either `20M` or `20m`.
Cache Timeout	The maximum number of seconds an entry remains valid in the cache.	600 seconds (10 minutes)	To allow entries to remain valid in the cache for 30 minutes, enter `1800`.

Exclude Web Application Files	A comma-delimited string that specifies the Web application files to exclude from caching. Enter the string in this form; items in brackets are optional:
	(<*WebAppName*>[/<*dir*>], [<*file_type*>], [<*file_type*>], ...), (<*WebAppName*>[/*dir*>], ...), ...
		An empty string	To exclude all the GIF and JPG files in the *images* directory and all the files in the *archives* directory for the Web application “Vacation”, enter:
			`(Vacation/images, *.gif, *.jpg), (Vacation/archives, *.*)`
			To exclude all the files in the Vacation Web application, enter:
			`(Vacation)`

Exclude Files	A comma-delimited string that specifies the non-Web application files to exclude from caching. Enter the file definitions in this form, relative to the document root directory:
	(<*dir*>, [<*file_type*>], [<*file_type*>], ...), (<*dir*>, [<*file_type*>], ...), ...
	Note By default, the document root directory is *$JAGUAR/html*. You can change it on the HTTP Config tab—see “HTTP Config” on page 25.
		An empty string	To exclude all the files in the *$JAGUAR/html/images* directory, enter:
			`(images, *.*)`
			To exclude all the files in the *$JAGUAR/html/images* directory and all the files in its subdirectories, enter:
			`(images)`

Flush Cache	Select to flush all the entries from the cache.	N/A	You can also flush the cache programmatically by calling the Management interface method flushStaticPageCache, which requires one string parameter. Currently, you must pass an empty string. In the future, multiple caches will require that you specify the cache name.

When you modify static page cache properties, you must refresh the cache for the changes to take effect. To refresh the cache, highlight the server icon and select File | Refresh Static Cache.

HTTP Custom Response Header

The HTTP Custom Response Header tab allows you to define custom response header filters for HTTP responses at the server level. You can customize header information such as the server name or the expiration date of the response. By default, a server-level custom response header filter applies to all server resources. You can apply the filter to specific resources by setting the value of the com.sybase.jaguar.server.filter-mapping property to the resource URLs.

You can also define custom headers at the Web application level by installing the default filter com.sybase.jaguar.servlet.AddHeadersFilter in a Web application. Chapter 23, "Using Filters and Event Listeners," in the *EAServer Programmer's Guide* describes how to do this. When both server and Web application custom headers exist, the Web application custom header takes precedence.

❖ **Defining custom headers**

Enter a custom header as property name/value pairs.

1 Click Add to display the New Property dialog box.

2 Enter a property name, property value, and (optionally) a description.

3 Select the property type, either String or Date. If you select Date, specify when the header expires. Enter a period of time, and select either From Now or Ago. For example, if you want the header to expire a month from now, enter 1 in the Months field, and select From Now.

To edit a header property, highlight the property and click Modify. Edit the property name or value, and click OK.

To delete a header property, highlight the property and click Delete.

For more information on filters and programming customized responses, see the Java Web page at http://java.sun.com/products/servlet/Filters.html.

HTTP Directory Browsing

The HTTP Directory Browsing tab allows you to enable HTTP directory browsing and define which directories can be browsed; these directories can be inside or outside Web applications.

You cannot enable directory browsing for the special Web application directories *WEB-INF* and *META-INF*. Also, if you enable browsing for a directory that contains welcome files, anyone browsing the directory will see only the welcome files.

❖ **Enabling directory browsing**

1 Click Enable Directory Browsing. This enables browsing for the directories you have added to your list.

2 Click Add. An entry is added to the list with default settings. Modify the entry to define the directory that can be browsed:

- Web Application Name – click the Web application name. A drop-down list displays a list of installed Web applications from which to choose. Or use Default to define non-Web application directories.
- Directory Name – enter the directory name that can be browsed.

  Note Use forward slashes when defining a directory name.

- Grant Access to Subdirectories – either true or false. If true, grants access to all subdirectories of the directories that you specify.

3 To allow browsing of multiple top-level directories in the same Web application, repeat the previous steps to create an entry for each directory. To delete a directory from the list, highlight the directory and click Delete.

4 Restart the server for the changes to take effect.

You can verify the settings of HTTP directory browsing by checking these properties on the Advanced tab:

- com.sybase.jaguar.server.http.dirbrowseenable must be set to true to enable directory browsing. If false, the following two properties are ignored.
- com.sybase.jaguar.server.http.dirbrowseinclude defines the list of non-Web application browsable directories.
- com.sybase.jaguar.server.http.dirbrowsewebappinclude defines the list of Web application browsable directories.

Components

Properties on this tab configure component execution and include:

- **LWC** Enables the EAServer lightweight container (LWC) for intercomponent EJB invocations or calls to EJBs from servlets and JSPs hosted in the same server. For more information, see "Lightweight container" in the *EAServer Performance and Tuning Guide*.
- **Skeleton Support** If LWC is enabled, the Skeleton Support option enables LWC calls to EJB components from servlets and JSPs hosted in the same server. Such calls are not supported unless this option is set.
- **External Server Request Timeout** For components that run in an external server, specifies the default external request timeout. That is, how long, in seconds, to wait for a response from the external server before returning an error to the client. If not set, the default is 60 seconds.
- **External Server Start Timeout** For components that run in an external server, specifies the default server start timeout. That is, how long, in seconds, to wait for the external server to start if it is not already running. EAServer returns an error to the client if the external server does not start in the specified time. If not set, the default is 60 seconds.

Advanced

For advanced users only. Select this tab to edit server property settings in the EAServer configuration repository. Properties are listed in Appendix B, "Repository Properties Reference." You can use this tab to edit any property prefixed with "com.sybase.jaguar.server." Most server properties can be configured on other tabs in the Server Properties dialog box, except the following—see the corresponding page numbers for information about each property:

- com.sybase.jaguar.server.authservice on page 507
- com.sybase.jaguar.server.authorization.service on page 506
- com.sybase.jaguar.server.authorization.permcachetimeout on page 506
- com.sybase.jaguar.server.bindrefresh on page 508
- com.sybase.jaguar.server.filter-mapping on page 516
- com.sybase.jaguar.server.http.disablechunkedtransfer on page 526
- com.sybase.jaguar.server.http.force.close on page 529
- com.sybase.jaguar.server.jvm.nojit on page 542
- com.sybase.jaguar.server.jvm.options on page 543

Configuring listeners

A listener is an EAServer port that communicates to clients using various protocols. For protocols that use SSL security features (HTTPS and IIOPS), you assign a security profile to the listener. The profile defines security characteristics of the listener. For protocols that do not use SSL (HTTP, IIOP, and TDS), no security profile is required.

This section describes the tasks required to configure listeners. You can:

- Create a new listener and assign a profile to it.
- Assign a profile to an existing listener.
- Modify listener settings for both secure (IIOPS and HTTPS) and unsecure protocols (TDS, IIOP, and HTTP).

Preconfigured listeners

EAServer comes with preconfigured listeners for all protocols. Secure protocols are assigned a predefined security profile.

The default settings for the preconfigured listeners are described in Table 3-9. Only secure listeners use security profiles.

Table 3-9: Default listener settings

Listener name	Port	Security profile
http	8080	
https1	8081	sample1
https2	8082	sample2
iiop	9000	
iiops1	9001	sample1
iiops2	9002	sample2
tds	7878	
OpenServer	7979	

The default host for these listeners is the name of the machine where the server is started. The host is defined in the setup script (*setenv.sh* or *setenv.bat*) using the JAGUAR_HOST_NAME environment variable. When you install EAServer, the installation process sets JAGUAR_HOST_NAME to the name of your machine. To change the host for a listener, use EAServer Manager or jagtool. For more information, see "Modifying an existing listener" on page 46 or the jagtool command set_props on page 288.

The OpenServer listener is intended for migrating existing Open Server™ applications to EAServer. See Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide* for more information.

Note You must restart the server for your changes to take effect. If you have changed the server's host name and port number, you must also restart EAServer Manager and reconnect to the server using the new host name and port number.

Listener failover

If a server cannot retrieve listener information from the repository for an IIOP listener or if an IIOP listener has not been configured, the server attempts to open a listener at this address:

```
IIOP: localhost, 9000
```

Listener start-up can fail if a port is already in use. You can verify the listener addresses in use by viewing the initial log entries in the *server_name.log* file (where *server_name* is the name of the server). If the log messages indicate a listener configuration problem, use EAServer Manager to connect to the indicated IIOP address and reconfigure the server's listener properties.

Creating and configuring listeners

This section describes how to create, modify, and delete a listener. All of the configuration tasks require you to first access the Listeners folder from EAServer Manager:

1 Double-click the Servers folder.

2 Double-click the server for which you want to create, modify, or delete a listener.

3 Click the Listeners folder on the left side of the window.

❖ Creating a new listener

1 Select File | New Listener.

2 Enter the name of the new listener, then click Create New Listener.

3 Complete the information in the Listener Info window. See Table 3-10 for the listener property descriptions.

The new listener appears on the right side of the window when you highlight the Listeners folder.

❖ Modifying an existing listener

1 Highlight the listener you want to modify.

2 Select File | Properties.

3 Make your modifications and click Save. Listener properties are described in Table 3-10.

Table 3-10: Listener profile properties

Property	Description	Comments
General/ Protocol	Select the protocol from the drop-down list: • HTTP • IIOP • TDS • HTTPS • IIOPS	HTTPS and IIOPS are secure protocols that provide all of the security features made available by SSL, including authentication and encryption. TDS, IIOP, and HTTP do not provide encryption. TDS and IIOP provide user name and password-based authentication.

Property	Description	Comments
General/ Host	The name or IP address of the EAServer host to which the listener is being assigned.	For predefined listeners, the value is set to the name of the machine where the server is started using the JAGUAR_HOST_NAME environment variable, which is defined in the setup script (*setenv.sh* or *setenv.bat*). You can also set the value to the IP address of the machine where the server is started using the JAGUAR_IP_ADDRESS environment variable. For servers that accept Internet connections, include the domain name to allow clients outside your domain to connect, as in www.trysybase.com. If your machine supports multiple network addresses, you can enter each host name or IP address in a list separated by commas. You can also enter the special value `0.0.0.0`, which causes EAServer to listen on all of the machine's host or IP addresses. When using multiple host addresses, EAServer creates a listener for each host on the specified port. **Note** In certain cases, a client may not be able to resolve a host name. For example, the client's DNS server or *hosts* file does not have an entry for the specified host. To avoid issues with such clients, specify the IP address in the listener configuration.
General/ Port	The port number on the host to which the listener is assigned.	Verify that the port is not in use by any other service.
General/ Jaguar Security Profile	Select one of the preconfigured security profiles from the drop-down list. This field is enabled for only the secure protocols (HTTPS or IIOPS).	You can create new security profiles that can be assigned to a listener. For more information, see "Configuring security profiles" in the *EAServer Security Administration and Programming Guide*.
General/ Enable Open Server Events	When selected, the TDS port accepts Open Server client connections; if not, only MASP requests are accepted.	You must use TDS as the protocol for Open Server events.
General/ Log SSL Errors	When selected, additional information about SSL errors is logged.	

Property	Description	Comments
General/ Connection Request Pool Size	For Solaris only. Configures the pool size for outstanding connection requests. Values must be a positive integer less than or equal to 4096. If this property is not set, the default is 128. Values greater than 4096 are truncated to 4096 to avoid excessive memory allocation at startup.	When the server is very busy, all available threads may be in use when a connect request arrives. These pending connect requests are pooled until they can be handled. If the pool size is too small, client connection requests may time out before the server can handle the request. You can configure different request pool sizes for different protocols. For example, if the server is handling mostly HTTP requests, you can increase the request pool size for the HTTP listener while leaving the IIOP request pool size at a low value. The connection request pool size affects the server memory requirements: `mem = entries * 20K` That is, each entry requires about 20K of memory reserved at server startup.
HTTP/ Keep Alive	For HTTP or HTTPS only. Specifies the time in seconds to keep open the server side of an idle HTTP connection.	If not specified, the default is 120.
HTTP// Maximum Requests	For HTTP or HTTPS only. Specifies the maximum number of HTTP requests to service before closing each connection.	If not specified, the default is 100.
HTTP// Enable Connector	For HTTP or HTTPS only. You must enable this option if you use the EAServer Web server redirector between HTTP clients and the EAServer HTTP listener.	**Warning!** This property must be set to true when clients connect via the Web server redirector. Otherwise, the HTTP response can be incorrect.
Resources/ Maximum Response Time	The maximum allowable average response time for each request, in seconds.	If the average response time rises above this limit, EAServer blocks additional connections until the average drops below the specified limit. The default is -1, which indicates no time limit.
Resources/ Minimum Number of Connections	When the Maximum Response Time is set to a non-default value, specifies the minimum number of clients that must be allowed to execute regardless of observed response times.	The default is -1, which means no new connections are blocked by the Performance Monitor. For more information on this feature, see Chapter 9, "Using the Performance Monitor," in the *EAServer Performance and Tuning Guide*.
Advanced	Allows you to configure properties in their "raw" format.	See Listener properties on page 456.

Using ports less than 1024 On UNIX platforms, if you configure listeners using port numbers less than 1024, you must start the server with a user ID that has root privileges. See “Changing the effective user ID of the server process” on page 54for more information.

❖ **Deleting a listener**

1 Highlight the listener you want to delete.

2 Right-click the listener and select Delete.

Starting the server

Before you can develop EAServer applications, you must start the server.

You can run a server in different modes, debug and normal, using different Java runtime versions and different Java virtual machines (VMs). To start the server in the desired configuration, change to the EAServer *bin* subdirectory and run either *serverstart.sh* (UNIX) or *serverstart.bat* (Windows) with the options described in Table 3-11.

Table 3-11: Server start options

Option	Description
-servername *server*	Specifies the name of the server; the default is “Jaguar”.
-jdk13 | -jdk14	Specifies the JDK version; the default is 1.3.
-IPV4 | -IPV6	Specifies which Internet protocol versions to support. -IPV4, the default, supports Internet Protocol Version 4 (IPV4). -IPV6 specifies support for IPV6 and IPV4. For more information, see “IPV6 support” on page 69.
-compilerversion [4x | 6x]	Solaris only. Specifies whether to run with libraries and binaries that are compatible with the version 6.*x* Solaris CC compiler format, or the version 4.*x* format. If not specified, the default is version 6.x compatibility.
-debug	Runs the server in debug mode.
-xterm	UNIX only. Runs the server in a new window opened by xterm.

Option	Description
-jvmtype [classic | client | server]	Specifies which Java VM to use: • Classic – classic Java VM • Client – HotSpot Client VM • Server – HotSpot Server VM If you run JDK 1.4, you can choose either of the HotSpot VMs; the default is the HotSpot Client VM. The classic VM is not supported with JDK 1.4. On Solaris, if you use JDK 1.3, you can run either of the HotSpot VMs. On Windows, if you run JDK 1.3, the default is the HotSpot Client VM.
-altthrdlib	On Solaris 2.8 systems, specifying this option causes EAServer to run with the alternate JDK thread library that uses one-to-one mapping between kernel threads and Java threads. This threading model can yield better performance and reliability than the default many-to-many threading model.
-jpdasuspend	When using the JPDA Java debugging option, you can specify this option to suspend the server at start-up time. This allows you to set break points in code that executes at start-up time, such as servlets that are configured to load on start-up. You can resume execution of the server with your Java debugger.
-msdev | -devenv	Windows only – enables in-process debugging in Microsoft Visual C++. • `-msdev` starts the server in Visual C++ version 5 or 6. • `-devenv` starts the server in Visual C++ version 7. The required Microsoft Visual C++ environment settings must be in effect for the server process. In *bin\user_setenv.bat*, set the VC variable to the Visual C++ installation, where *vc_path\bin* is the location of the *vcvars32.bat* file: `set vc=c:\vc_path`

-workshop	Solaris only – enables in-process debugging in the Workshop debugger. Define the location of the Workshop debugger by setting the WORKSHOP_DIR variable in the *user_setenv.sh* file; for example:
	`WORKSHOP_DIR=/OPT/SUNWspro6.2/`
	`export WORKSHOP_DIR`
	For information about the Sun Workshop debugger, see the Workshop Web page at http://www.sun.com/990209/workshop/.

Option	Description
-install | -uninstall | -remove | -removeandinstall | -stop | -start | -c	Windows only – these options apply to servers that run as Windows services: • `-install` installs the server as a Windows service. • `-uninstall` and `-remove` uninstall the server from the list of Windows services. • `-removeandinstall` removes the server from the list of Windows services, then reinstalls it. • `-stop` shuts down a server that has been installed as a Windows service. You can also stop the server using the Services dialog in the Control Panel, or from EAServer Manager. • `-start` starts a server that has been installed as a Windows service. When installed as a service, the server starts automatically when Windows starts. Use this option only after you have stopped or shut down the server. • `-c` (the default) starts the server as a console application. Use this option if you have configured the server to run as a Windows service, but you want to run it in a console.
-servicename *service*	Windows only – if running or installing a server as a Windows service, specifies a service name. If not specified, the default is the server name.
-v	Windows only – displays version information in the console, then exits.

"Operating system configuration" on page 73 describes how to set environment variables that you may need to run the server.

❖ **Starting a server in UNIX**

1 Add the location of the X-Windows *xterm* utility to your *path* variable. For example:

```
set path = ($path /usr/local/SUNWmotif/bin/)
```

2 Edit the *$JAGUAR/bin/setenv.sh* shell script. Change the JAGUAR variable to the EAServer home directory, then run the *serverstart.sh* script, using this syntax:

```
serverstart.sh [-servername server] [-jdk13] [-jdk14] \
  [-debug] [-xterm] | [-jvmtype classic | client | server]
  [-workshop]
```

See Table 3-11 on page 49 for a description of the server options.

Debug servers Debug servers allow you to remotely debug components from tools that support EAServer component debugging, such as PowerBuilder or PowerJ. You cannot run the debug server unless you installed the debug libraries and binaries. The debug server cannot run as a Windows service.

Running servers in the background

You should run production servers in the background with the nohup command. This allows you to log out of the system without shutting down the server.

To run a server in the background, change to the *$JAGUAR* directory, and run this command:

```
nohup ./serverstart.sh -servername server > serverconsole.log 2>&1 &
```

where *server* is the server name as displayed in EAServer Manager. nohup runs the server and prevents its shutdown when you log out. Console output (stderr and stdout) is directed to a file named *serverconsole.log*.

❖ **Starting a server in Windows**

- Change to the *%JAGUAR%\bin* directory, and run *serverstart.bat* using this syntax:

```
serverstart [-servername server] [-jdk13 | -jdk14] [-debug] \
  [-msdev | -devenv] [-jvmtype classic | client | server] \
  [-install | -uninstall | -remove | -removeandinstall | \
   -stop | -start | -c] [-v]
```

See Table 3-11 on page 49 for a description of the server options.

Starting servers that are installed as Windows services

If a server is installed as a Windows service, the server is started automatically when you start Windows. To start a service manually, use the Services dialog in the Windows Control Panel.

On Windows 2000 and Windows XP:

1 Select Start | Settings | Control Panel | Administrative Tools | Services.

2 In the list of Services, find the name of your server (for example, the preconfigured server is “Jaguar”).

To stop the server, double-click the server name, and in the server Properties dialog box, click Stop.

To start the server, double-click the server name, and in the server Properties dialog box, click Start.

You can also run this command in the EAServer *bin* subdirectory:

```
serverstart -servername service_name -start
```

where *service_name* is the name of the server. To stop the server, use:

```
serverstart -servername service_name -stop
```

Installing servers as Windows services

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

❖ **Installing a server as a Windows service**

1 If the server is not running, start it on the desktop.

2 Start EAServer Manager and connect to the server using EAServer Manager.

3 Configure any environment variables required for the server process. The JAGUAR, PATH, CLASSPATH, BOOTCLASSPATH, and BOOTLIBRARYPATH environment variables are saved in the Windows Registry when you install the server as a service. These variables keep the values they have when the server is installed as a service. You must set other variables in the server properties file, using the Advanced tab to specify values as described in Environment variable properties on page 562. Settings in the start-up scripts (*serverstart.bat*, *setenv.bat*, and *user_setenv.bat*) are not applied when the server starts as a service.

4 Shut down the server by highlighting its icon and choosing File | Shutdown.

5 Run the following command in the EAServer *bin* subdirectory, specifying the server name in place of *server* and the JDK version string in place of *jdk*:

```
serverstart -servername server -jdk -install
```

JDK version	JDK version string
1.4	jdk14
1.3	jdk13

For debugging, you can use the following options when you install the server:

Option	Description
-servicesleep *n*	Causes the service to sleep *n* seconds during start-up. Set *n* to a value large enough to allow you time to connect to the process.

Option	Description
-svclog	Specifies the full path and name of a log file to capture debug information about the service; installing, starting, stopping, and running.

❖ **Removing a server from the list of Windows services**

- Run the command:

```
serverstart -servername servicename -remove
```

where *servicename* is the name of the server as displayed in the Services dialog box.

Edit service names separately If you delete a server definition in EAServer Manager after installing a Windows service for that server, run the serverstart command to remove the server from the list of services.

Determining server version level

Run the following command from the EAServer *bin* subdirectory:

```
serverstart -servername server -v
```

This command displays the version of the server, without starting the server.

Changing the effective user ID of the server process

EAServer on UNIX platforms allows you to configure an effective user name and group for the server to run as. This feature is useful if you start the server while logged in as a UNIX user with administrator privileges: you can start the server with administrator privileges, but the server switches to an account that has fewer privileges before it begins accepting client connections. When changing the effective user that runs the process, you must use a group name that the effective user is a member of. If not, the error `Invalid OS group specified: 'groupname'` is generated in the EAServer log file. For example, if you set username to user1 and groupname to group1 and start the server as user2, an error is generated if user2 is not a member of group1. To change the effective account, set the following properties in the All Properties tab in the EAServer Manager Server Properties dialog box, or with jagtool:

- com.sybase.jaguar.server.unix.username specifies the user name to switch to.
- com.sybase.jaguar.server.unix.groupname specifies the group name to switch to.

This feature is useful if you use listener ports less than 1024, such as 80 for HTTP and 443 for SSL. Port numbers less than 1024 cannot be used unless the server is started by the root user. After establishing network listeners, the server switches to the specified user and group. This allows you to start the server with listeners using standard HTTP and SSL port numbers, while running it as an account that has fewer privileges.

These settings are ignored on Windows platforms.

Using the JagRepair server

You can use the JagRepair server when you cannot start your server; for example, if you have specified incorrect Object Transaction Service (OTS) settings which prevent the server from starting and makes it impossible to correct the problem. The JagRepair server is read-only and provided for repair purposes only.

❖ **Starting and connecting to the JagRepair server**

1 Change to the EAServer *bin* subdirectory and enter:

- On UNIX:

```
serverstart.sh -servername JagRepair
```

- On Windows:

```
serverstart -servername JagRepair
```

2 Start EAServer Manager.

3 Select Tools | Connect | EAServer Manager and enter:

- `jagadmin` as the User Name
- <*host_machine_name*> as the Host Name
- `9000` as the Port Number

4 Click Connect.

Configuring log profiles

A server's logging properties are defined in a Log Profile, which defines the logging subsystem used as well as other properties, such as output destinations, formats, and the level of severity required before a message is recorded. You can also configure different log profiles for the debug and production server versions.

EAServer supports industry-standard Java logging APIs, and provides APIs to log messages from components of other types. You can use the following logging subsystems:

- The built-in EAS subsystem, which offers the same functionality available in EAServer 4.x versions, plus several enhancements including:
  - The ability to configure log levels so that messages below a specified level of severity are discarded.
  - Support for different logging configurations in the debug and production servers.
  - Optional archiving and compression of previous log file versions.
  - More control over message formatting.
- Apache Log4j, which is commonly used on large projects. For more information on this package, see the Apache Log4j Documentation at http://jakarta.apache.org/log4j/docs/api/overview-summary.html.
- The Java Logging package, included in JDK 1.4. This API is the Sun proposed standard for logging in Java applications. For more information, see the Java Logging documentation at http://java.sun.com/j2se/1.4.1/docs/guide/util/logging/overview.html. To use this package, your server must be running JDK 1.4 or later.

If you use Log4j or Java Logging, you can extend default behavior by plugging in your own code that implements the required interfaces. For example, you can install Log4j log handler classes that write messages to the Windows event log or to a database. Also, if you want one of these packages to log messages from your own component or application code, you can configure the server's log profile so that server log messages go to the same destinations. In addition, if you use Log4j or the Java Logging system, you can log messages from in-server Java code by calling the logging API directly.

Using legacy logging methods Regardless of the logging system you use, you can write messages to the log using all of the methods supported in earlier versions of EAServer, such as:

- System.out.println or Jaguar.writeLog from Java code running in the server
- ErrorLogging.log from PowerBuilder NVO components
- JagLog from C or C++ components
- IJagServer.writeLog from ActiveX components

Log output from these legacy methods can be configured using the srv_log logging category.

Configuring the server's log profile

To change the log profile used in a server, display the Log/Trace tab in the Server Properties dialog box. To change the profiles using jagtool, set the following server properties:

- com.sybase.jaguar.server.logging.profile.debug specifies the name of the log profile used in the debug server.
- com.sybase.jaguar.server.logging.profile.prod specifies the name of the log profile used in the production server.

EAServer installs several predefined log profiles, as listed in Table 3-12.

Table 3-12: Preconfigured log profiles

Profile name	Description
debug	Default logging profile for debug servers. Supports the logging mechanism used in versions of EAServer earlier than 5.0.
prod	Default logging profile for production servers. Supports the logging mechanism used in versions of EAServer earlier than 5.0.
debug_jdk	Supports the JDK 1.4 java.util.logging package, configured for use in debug servers. The server must be started with JDK 1.4 to use this profile.
prod_jdk	Supports the JDK 1.4 java.util.logging package, configured for use in production servers. The server must be started with JDK 1.4 to use this profile.
debug_l4j	Supports Log4j, configured for use in debug servers.
prod_l4j	Supports Log4j, configured for use in production servers.

Configuring log profiles

You can configure log profiles in EAServer Manager or in a text editor.

❖ **Defining a new log profile**

The easiest way to define a new log profile is to copy one of the existing definitions to a file, edit the file, then import the definition in Jaguar Manager as follows:

1 Copy the predefined log profile that is closest to the characteristics that you want, specifying a new file name with the *.props* extension.

2 Open the file in a text editor, and change the value of the com.sybase.jaguar.logprofile.name to match the base name. You can edit the remaining properties after importing the profile, using the Log Profile Configuration wizard.

3 Import the profile to EAServer Manager as follows:

   a Start EAServer Manager if it is not already running.

   b Highlight the Log Profiles folder in the left pane and choose File | Import.

   c Specify the file name you created and click OK.

   d You should see the new profile in the Log Profiles folder.

You can also create a log profile using the New Log Profile wizard.

❖ **Creating a log profile using the New Log Profile wizard**

- In EAServer Manager, highlight the Log Profiles folder, and select File | New Log Profile Wizard.

  The wizard guides you through the process of creating a log profile.

❖ **Configuring log profiles**

To configure log profiles, you can use either the Log Profile Configuration wizard or the Log Profile Properties page. The Log Profile Configuration wizard guides you through the basic steps required to configure a log profile for an EAS, Java Logging, or Log4j logging subsystem.

1 From within EAServer Manager, display the list of log profiles by expanding the Log Profiles folder.

2 Highlight the log profile you want to configure.

3 Select File | Configuration Wizard or File | Properties.

Note To find property descriptions for Log4j and Java Logging subsystems, see:

- Log profile Log4j subsystem properties on page 475
- Log profile Java Logging subsystem properties on page 472

---

4 To change the other properties, double-click the log profile name in the left pane to display the following:

- *Categories*, which map to the various subsystems of EAServer. Different subsystems can use different category names for log messages, and you can configure the output for each category. Table 3-13 lists the category names used in EAServer. To change these properties:
  - Highlight Categories in the left pane to display the logging categories in the right pane.
  - Highlight the category name in the right pane and choose File | Configuration Wizard. Edit the properties described in "Category properties" on page 63.

  You can also add new categories by highlighting Categories in the left pane and choosing File | New Log Category Wizard.

- *Handlers*, which define how messages are logged. Associating a handler with a category determines how messages from that category are logged. To change these properties:
  - Highlight Handlers in the left pane to display the handler names in the right pane.
  - Highlight the handler name in the right pane and choose File | Configuration Wizard. Edit the properties described in "Handler properties" on page 64.

  You can also add new handlers by highlighting Handlers in the left pane and choosing File | New Log Handler Wizard.
- *Formatters*, which specify a format pattern for messages. To change these properties:
  - Highlight Formatters in the left pane to display the formatter names in the right pane.
  - Highlight the formatter name in the right pane and choose File | Configuration Wizard. Edit the properties described in "Formatter properties" on page 66.

  You can also add new formatters by highlighting Formatters in the left pane and choosing File | New Log Formatter Wizard.

❖ **Exporting log profiles**

You can export log profile settings to a file. This is useful when you want to use the settings to control logging from Java client applications as described in "Using log profiles in Java client applications" on page 67. Export log profile settings as follows:

1 Highlight the Log Profile and Choose File | Export.

2 Specify the directory to export the file to, then choose Save. EAServer Manager creates *profile.props* in the specified directory, where *profile* is the name of the logging profile.

Category names

Category names map to the various subsystems of EAServer. Different subsystems can use different category names for log messages, and you can configure the output for each category. Table 3-13 lists the category names used in EAServer. Many of these are not used in the preconfigured log profiles, but you can add them.

❖ **Configuring a log category using the Log Category Configuration wizard**

The Log Category Configuration wizard guides you through the process of configuring a log category.

1 From within EAServer Manager, display the list of log profiles by expanding the Log Profiles folder.

1 Expand a log profile, then expand the Categories folder.

2 Highlight the log category you want to configure.

3 Select File | Configuration Wizard.

Table 3-13: Log category names used in EAServer

Category name	Used by
<root>	N/A. This category is used as the default parent for categories for which a parent category is not specified.
axdisp	The ActiveX component dispatcher.
com.sybase	All EAServer internal subsystems that do not specify another name.
com.sybase.CORBA	CORBA component dispatcher systems that do not use another name.
com.sybase.CORBA.cts	Parent category for the following categories that use this prefix in the name.
com.sybase.CORBA.cts.HeapStorage	The *CtsComponents/HeapStorage* storage component that manages in-memory replication of EJB session bean state and HTTP session state.
com.sybase.CORBA.cts.JdbcStorage	The *CtsComponents/JdbcStorage* storage component that manages database storage of EJB CMP entity bean data, EJB session bean state, and HTTP session state.
com.sybase.CORBA.cts.MessageService	The EAServer message service.
com.sybase.CORBA.cts.MessageService.network	The EAServer message service, for debug messages related to network connectivity (logged when the cms.debug.network message service property is set to true).
com.sybase.CORBA.cts.MessageService.session	The EAServer message service for messages related to session connectivity (logged when the cms.debug.session message service property is set to true).
com.sybase.CORBA.cts.ObjectCache	The object cache, which creates and manages the caches used for servlet and JSP response caching and entity component query and instance caching.
com.sybase.CORBA.cts.ThreadManager	The thread manager, which provides an API to run components asynchronously.
com.sybase.CORBA.iiop	The Java client runtime IIOP subsystem, for logging of errors.
com.sybase.CORBA.iiop.giop	The Java client runtime IIOP subsystem, for logging of General Inter-Orb Protocol (GIOP) trace data (if GIOP tracing is enabled).

Category name	Used by
com.sybase.CORBA.iiop.iiop	The Java client runtime IIOP subsystem, for logging of IIOP trace data (if IIOP tracing is enabled).
com.sybase.CosNaming	The EAServer naming service.
com.sybase.CosNaming.console	The EAServer naming service, to output messages to the console during server initialization. (You can override the default destination so these messages do not go to the console.)
com.sybase.ejb.lwc	The EJB component lightweight container.
com.sybase.jaguar.deployment	The server-side deployment subsystem.
com.sybase.jaguar.importer	The client-side deployment subsystem (used when importing J2EE archives, Java class and interface definitions, and ActiveX interfaces among other things).
com.sybase.jaguar.jcm	The Java connection cache manager.
com.sybase.jaguar.jcm.cm	The Java connection cache manager, for messages related to connector management.
com.sybase.jaguar.jcm.sm	The connection cache manager, for messages related to security.
com.sybase.jaguar.management	Internal configuration management Java classes.
com.sybase.jaguar.packager	The deployment subsystem, when exporting J2EE archives.
com.sybase.jaguar.server	Server subsystems that don't use another name.
com.sybase.jaguar.server.cl	Server Java class loader.
com.sybase.jaguar.server.cl.copy	Server Java class loader, when creating temporary copies of JAR files.
com.sybase.jaguar.server.hsb	Server hot standby system.
com.sybase.jaguar.servlet	The EAServer servlet engine.
com.sybase.jaguar.servlet,cl	The servlet engine class loader.
com.sybase.jaguar.servlet.hsc	The servlet engine HTTP session context manager.
com.sybase.jaguar.servlet.security	The servlet engine security manager.
com.sybase.jms	The message service JMS API implementation.
libjagcm	Native connection cache manager.
libjcc	The C++ client runtime implementation (used by C++, PowerBuilder, and ActiveX clients as well as Java application clients that use the EAServer SSL implementation).
libjdispatch	Native component dispatcher code.
libjdispatch.iiop	Server-side IIOP protocol handler code.
libjdispatch.iiop.read	Server-side IIOP protocol handler code, read handling.
libjdispatch.iiops	Server-side IIOPS protocol handler code.
libjdispatch.management	Native code called from the management API (for example, when restarting or refreshing the server).
libjdispatch.monitoring	Native code called from the monitoring API.

Category name	Used by
libjdispatch.nip	Named instance pool implementation.
libjdispatch.nipmgr	Named instance pool manager.
libjdispatch.transaction	Native transaction manager code.
libjhttp.dynamo	PowerDynamo integration subsystem.
libjhttp.log	Native in-server HTTP protocol handler code.
libjhttp.statlogger	Native in-server HTTP protocol handler code, statistics provider implementation.
libjjdk12	The server subsystem that manages the creation and configuration of the in-server Java VM.
libjots	Server-side object transaction service implementation.
srv_log	Output from the legacy logging methods such as: • System.out.println or Jaguar.writeLog from Java code running in the server • ErrorLogging.log from PowerBuilder NVO components • JagLog from C or C++ components • IJagServer.writeLog from ActiveX components

Category properties

Table 3-14 describes the category settings.

Table 3-14: Logging category settings

Tab name / Setting name	Specifies
General / Description	An optional description.
Options / Level	The logging level. Only messages of equal or greater severity are logged. Table 3-15 on page 64 describes the severity codes.
Options / Use Parent Handlers	Whether to log to the parent category handlers as well as the handler associated with this category.
Options / Parent	The parent category.
Options / Resource Bundle	The name of a Java resource bundle containing localized messages that are logged from Java code. If this property is not set, the default resource bundle is the class ResourceBundle in the package with the same name as the category. For example, the default resource bundle for the category com.sybase.jaguar.server is com.sybase.jaguar.server.ResourceBundle.
Handler	One or more logging handler names. The handler properties specify how and where messages are logged. See “Handler properties” on page 64 for more information.

Table 3-15 lists the severity level codes. You can set the logging level in the category to discard messages below a specified severity level.

Table 3-15: EAS log subsystem error levels

Level	To log
ALL	All messages.
CONFIG	Configuration errors. You should correct these problems.
ERROR	Errors that prevent completion of a requested action. For example, a component has thrown an uncaught exception and its transaction is being rolled back.
FATAL	Errors that indicate the server should terminate.
DEBUG	Debug messages.
FINE	Debug messages.
FINER	Has same effect as FINE.
FINEST	Has same effect as FINE.
INFO	Informational or status messages. For example, the name server has finished binding components.
OFF	No messages are logged.
WARN	Warning messages. For example, a warning might be that the server is in a cluster and other members are not found.

Handler properties

Handlers specify how and where messages are logged. You can associate handlers with categories to configure the output of messages from that category. Table 3-16 on page 64 describes the handler settings.

❖ **Configuring a log handler using the Log Handler Configuration wizard**

The Log Handler Configuration wizard guides you through the process of configuring a log handler.

1 From within EAServer Manager, display the list of log profiles by expanding the Log Profiles folder.

1 Expand a log profile, then expand the Handlers folder.

2 Highlight the log handler you want to configure.

3 Select File | Configuration Wizard.

Table 3-16: Handler settings

Tab name / Setting name	Specifies
General / Description	An optional description.

Tab name / Setting name	Specifies
Type / Type	The output type. Choose one of the following: • file – outputs to a text file. • console – outputs to the console (standard output or standard error). • null – no output.
Type / File Name	When logging to a file, the pattern string to name the output file. You can use the placeholders in Table 3-17 on page 65.
Type / Truncate Log File	Whether to truncate the log file when the server starts. If truncation is not enabled, log messages from the server process are appended to those from the previous process. With truncation enabled, you can keep the previous log contents by enabling archiving (recommended) or rotation.
Type / Max Size	The maximum allowed size before archiving or rotating the log file. Use one of the following values: • `-1` – to specify no limit. • ***n***`m` – to specify *n* megabytes (with *n* a positive integer). • ***n***`k` – to specify *n* kilobytes (with *n* a positive integer).
Type / Max Time	The maximum amount of time before archiving or rotating the log file; enter as HH:MM.
Type / Rotate at Max	Whether to rotate log files at start-up and when the maximum size or time is reached. If you enable rotation, the log file is renamed with a sequential numeric extension when a new file is started. For example, if the file is *Jaguar.log*, previous versions are named *Jaguar.log.1*, *Jaguar.log.2*, and so forth.
Type / Archive Log Files	Whether to archive log files at start-up and when the maximum size is reached.
Type / Archive File Name	If archiving is enabled, the pattern string to name the archived log files. You can use the placeholders in Table 3-17 on page 65.
Type / Compress Archive File	If archiving is enabled, whether to compress the archived files. If you enable compression, the resulting archive name is the specified archive file name plus the ".zip" extension.
Type / Console Type	When writing to the console, the destination device. Choose stdout for standard output, and stderr for standard error.
Type / Host Name	When writing to a socket, the host name to connect the socket to.
Type / Port	When writing to a socket, the port number to connect the socket to.

When specifying log file names and archive file names, you can use the placeholders in Table 3-17.

Table 3-17: Log file name and path placeholders

Placeholder	Specifies
`${JAGUAR}`	The EAServer installation directory (specified by the JAGUAR environment variable).

Placeholder	Specifies
`${JAGEXEDIR}`	The location of binaries in the EAServer installation (*$JAGUAR/bin* on most platforms).
`${JAGSRVNAME}`	The server name, as displayed in EAServer Manager.
`%t`	The time of day when the file was created, formatted as HHMMSS.
`%d`	The date (day of month, month, and year) when the file was created, formatted as DDMMYYYY.

Formatter properties

Formatters specify the format of logged messages. You can associate a formatter with a handler to specify the format of messages logged through the handler. Table 3-18 describes the formatter properties.

❖ **Configuring a log formatter using the Log Formatter Configuration wizard**

The Log Formatter Configuration wizard guides you through the process of configuring a log formatter.

1 From within EAServer Manager, display the list of log profiles by expanding the Log Profiles folder.

1 Expand a log profile, then expand the Formatter folder.

2 Highlight the log formatter you want to configure.

3 Select File | Configuration Wizard.

Table 3-18: Formatter settings

Tab name / Setting name	Specifies
General / Description	An optional description.
Type / Message Format	The format pattern for the message text, using the placeholders listed in Table 3-19. You can use the Insert Tag control to insert these placeholders, or type them yourself.
Type / Insert Tag	When editing the message format, use this control to insert the placeholder tags.
Type / Simple Date Format	The format for timestamps in log messages. The pattern for the timestamps embedded in log messages, as converted to strings by java.text.SimpleDateFormat. For details, see the DateFormat API documentation at http://java.sun.com/j2se/1.4.1/docs/api/java/text/SimpleDateFormat.html.

Table 3-19 lists the placeholders that you can use when specifying the message format pattern. Each placeholder represents part of the message.

Table 3-19: Message format placeholders

Placeholder	Represents
%LN	The logging category name
%MC	Message code (number)
%ML	Message level (severity)
%MT	Message text
%PT	Processed message text, which consists of the message number (if specified when the message was logged) and the message text
%SN	Sequence number
%SF	Source file name
%SL	Line number in the source file
%SM	Method name in the source file
%TI	Thread ID (enabled by setting server property com.sybase.jaguar.server.logspid)
%TH	Exception thrown (if available)
%TS	The timestamp, formatted as specified by the Simple Date Format setting
%NL	A line break

Using log profiles in Java client applications

The EAServer Java client runtime classes also use the EAServer logging system. By default, the log level is ERROR and messages are sent to the console. You can change the client logging settings by specifying a configuration file. The same procedure works for Java/CORBA and EJB clients, as well as EAServer Manager and jagtool.

❖ **Specifying log settings for Java client applications**

1. Use EAServer Manager to create a log profile with the desired settings and export the settings to a properties (*.props*) file.
2. When running the application, specify the logging properties to use by setting the Java system property com.sybase.jaguar.logger.config.file to the path and file name of the properties file. For example, to specify the system property on the command line, use syntax like this:

```
java -Dcom.sybase.jaguar.logger.config.file=mylog.props \
   com.mycompany.MyClient
```

Configuring server stack size

EAServer has a stack size property that determines the amount of memory reserved for the call stack associated with each thread created by the server. EAServer runs each client request on a different thread, so the stack size is the dominant factor in determining how many client requests can be served simultaneously.

The default stack size is 256K on UNIX systems and 32-bit Windows operating systems. This is appropriate for almost all situations, and provides adequate reserve memory for the largest case loads that have been tested by Sybase engineering and customers.

For production servers that see heavy use from large numbers of clients, you may want to decrease the stack size from the default value. However, you must ensure that the stack size is adequate for the components running on the server. If the stack size is too small, your server may experience thread stack overflow errors, which are recorded in the server log.

Sybase recommends that you do not reduce the stack size if you run:

- Components that call third-party DLLs or shared libraries
- Java components that call native classes (including JDBC drivers that call out to native libraries)

Warning! Do not reduce the stack size below 32K. If you reduce the stack size, test your server thoroughly under heaviest client loads and check the log for stack overflow errors.

There are different procedures for setting the stack size on UNIX and Windows platforms.

❖ **Configuring stack size for servers running on UNIX**

1 Highlight the icon for the server and select File | Properties.

2 Display the Resources tab. You may need to scroll to the right to see this tab.

3 Enter a stack size in the Thread Stack Size field, specified in bytes as a decimal number. (The field displays with no value if you have not specified a value before. This means the default setting is in effect.)

4 Stop and restart the server.

❖ **Configuring stack size for servers running on Windows**

To change the thread stack size, you must have the Microsoft editbin utility, which is included with Microsoft Visual C++. This command line utility allows you to modify object files, executable files, and dynamic link libraries (DLLs). For more information on the editbin utility, see the Microsoft Visual C++ documentation.

1 Save a copy of the original server executable, *jagsrv.exe* in the EAServer *bin* subdirectory.

2 Change the thread stack size by running the following command in the EAServer *bin* subdirectory:

```
editbin /stack:value jagsrv.exe
```

where *value* is the new stack size, specified in bytes as a decimal number. editbin rounds the value up to the closest number divisible by four. For example, the following command sets a 64K stack size:

```
editbin /stack:65536 jagsrv.exe
```

3 You can confirm the new setting by running the Microsoft dumpbin utility, which is included with Microsoft Visual C++. Run the following command:

```
dumpbin /headers jagsrv.exe
```

In the output, the stack size appears as a hexadecimal number on a line such as this:

```
 80000 size of stack reserve
```

IPV6 support

EAServer supports the Internet Protocol Version 6 (IPV6) on platforms that provide the underlying network support such as Windows 2003, Windows XP, and Solaris 2.8. Windows 2000 does not support IPV6. IPV6 support also requires JDK 1.4, or a later JDK version.

Server support for IPV6

By default, the server supports IPV4 connections only. To start the server with support for IPV6, specify the -IPV6 and -jdk14 command line options. With these options, the server will accept both IPV4 and IPV6 connections. For more information, see "Starting the server" on page 49.

For servers that run in clusters, you must also set the server property com.sybase.jaguar.server.cluster.IPV6serverID using jagtool or the Advanced tab in the EAServer Manager Server Properties dialog box. For each server, set this property to a string that simulates an IPV4 address that is unique among the servers in the cluster; for example:

```
10.123.456.789
```

Client support for IPV6

To support IPV6, Java clients must run in a version 1.4 or later Java virtual machine. C++ clients and other clients that use the C++ runtime (such as PowerBuilder) must run with the required system network libraries in the DLL or shared library search path.

IPV6 host addresses can contain colons. To embed IPV6 host addresses in IIOP or HTTP URLs, surround the host address with square brackets as specified by RFC 2732, Format for Literal IPv6 Addresses in URLs at http://www.ietf.org/rfc/rfc2732.txt. For example:

```
http://[fe80::2c0:4fff:fe79:858d]:8080/
iiop://[fe80::2c0:4fff:fe79:858d]:9000/
```

Using Admin mode

You can run servers in Admin mode to perform maintenance on the server without allowing connections from your application end users. In *Admin mode*, the server accepts connections from EAServer Manager, jagtool, and clients that set the admin ORB option (described further below). A server that is accepting regular client connections is said to be in *Ready mode*.

In some cases, a server may switch to Admin mode when starting if a configuration problem prevents normal operation. For example:

- The server is in a cluster, and its cluster version and start-up mode properties indicates that the configuration is not synchronized with the master configuration. In this case, you must synchronize the cluster as described in “Synchronization” on page 131.
- The message service is installed, but cannot start for some reason such as failure to connect to the remote database server that stores message data. In this case, you must correct the configuration problem as described in Chapter 8, “Setting up the Message Service.”

When the server starts in Admin mode, or switches to Admin mode, it logs a message like this:

```
The server is in "admin" mode (reason).
```

where *reason* is text that describes why the server is in Admin mode. You see a similar message in EAServer Manager when you connect to a server running in Admin mode. With jagtool, you can check whether the server is in Admin mode with the getserverinfo command. In a custom management client, you can call the Jaguar::Management::getStatus() method: a return value of “ready” indicates normal operation. Any other value is text describing the reason the server is in Admin mode.

Putting servers into Admin mode

You can manually switch servers to Admin mode using any of the following techniques:

- Using jagtool and running the set_admin and restart commands.
- Using a custom management client that calls the setAdmin and restart methods in the Jaguar::Management API.

The server must be restarted to run in Admin mode. You can specify your own reason when placing the server in Admin mode.

Switching to Ready mode

You can switch a server from Admin mode to Ready mode using any of the following techniques:

- Using EAServer Manager, by connecting to the server, highlighting the server icon, and choosing File | Set Ready.

- Using jagtool, by running the set_ready commands.
- Using a custom management client, by calling the setReady method in the Jaguar::Management API.
- By deleting the file *<servername>.admin* in the *Repository/Server* subdirectory of the EAServer installation, where *<servername>* is the server name and restarting the server.

Unless you manually delete the *<servername>.admin* file, the server does not require a restart to switch to Ready mode; it begins accepting regular client connections immediately. If the configuration that you performed requires a server restart, do it before switching to Ready mode.

Creating clients that connect to Admin mode servers

A server that is in Admin mode refuses connections from normal clients. To create clients that connect to the server running in Admin mode, you must set the admin ORB option for your client type as described in Table 3-20.

Table 3-20: Setting the admin ORB option

Client type	How to set the property
Java/CORBA	Set the com.sybase.corba.admin option to true.
EJB	Set the com.sybase.ejb.admin option to true.
C++, PowerBuilder, or ActiveX	Before running the program, set the environment variable JAG_ADMIN to true, or set the -admin ORB option to true by passing it on the command line (for ActiveX or C++ only) or by setting it in the ORB initialization options.

Admin mode clients can call any components installed in the server, subject to the standard role-based security restrictions. However, the server accepts Admin mode connections before all normal initialization has completed. Therefore, keep in mind the following caveats when creating or running Admin mode clients:

- Service components may not have finished starting. You can check the status of installed services using the jagtool getservicestate command or by reading the service status messages in the server log file.
- If the server is in a cluster, it may not have joined the cluster.

Operating system configuration

❖ **Configuring environment variables**

1 Create a *user_setenv.sh* (UNIX) or *user_setenv.bat* (Windows) file in the EAServer *bin* subdirectory.

2 Edit the file and set environment variables such as CLASSPATH, BOOTCLASSPATH, or BOOTLIBRARYPATH here. The file is read when you start the server.

❖ **Setting the BOOTCLASSPATH and BOOTLIBRARYPATH variables**

When you are running EAServer with JDK 1.3 or later, you may need to edit the BOOTLIBRARYPATH and BOOTCLASSPATH environment variables. You can set these variables in the *user_setenv.sh* or *user_setenv.bat* file.

1 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.

2 Set BOOTCLASSPATH if your EAServer components require Java classes that are not in the standard locations (the *html/classes* or *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 Chapter 11, “Creating CORBA Java Components,” and Chapter 21, “Creating Web Applications,” in the *EAServer Programmer’s Guide* for more information.

❖ **Setting the JAGUAR_RANDOMSEED variable**

EAServer 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 UNIX machines, EAServer accesses the process, virtual memory, and network statistics. On Windows, EAServer accesses the contents of the HKEY_PERFORMANCE_DATA Registry entry. If this Registry entry does not exist on your machine, you do not need to add it. You can set the JAGUAR_RANDOMSEED variable to improve EAServer 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, EAServer 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, EAServer 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 EAServer.

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

3 Restart the server.

CHAPTER 4

Database Access

This chapter describes the databases to which EAServer components can connect, and configuration tasks that you can perform to customize your environment, such as defining new connection caches and J2EE connectors, and configuring Adaptive Server Enterprise user connections.

You can perform most configuration tasks using EAServer Manager.

For information on configuring EAServer naming services, see Chapter 5, “Naming Services.” For information on configuring all aspects of EAServer security, including establishing and changing listeners, native SSL support, using digital certificates, roles, Web application security, and so on, see the *EAServer Security Administration and Programming Guide.*

Connecting to databases

This section contains details about connecting EAServer components to specific third-tier databases, including:

- Sybase Adaptive Server Anywhere
- Sybase Adaptive Server Enterprise and gateways
- Oracle databases
- Other databases

Sybase Adaptive Server Anywhere

On most platforms, EAServer includes a copy of Sybase Adaptive Server Anywhere (formerly called Sybase SQL Anywhere). The EAServer sample components use a preconfigured Adaptive Server Anywhere database and Adaptive Server Anywhere ODBC data source. The sample C++ components connect using an ODBC connection cache. The sample Java components connect using the Sybase jConnect™ for JDBC™ driver. View the predefined sample connection caches in EAServer Manager for examples of configuring a cache to connect to Adaptive Server Anywhere.

EAServer includes the jConnect runtime classes, and you can connect to Adaptive Server Anywhere with the same settings that you would use for Adaptive Server Enterprise. See “Sybase Adaptive Server Enterprise and gateways” on page 76.

Do not use the Sun JDBC-ODBC bridge driver Some EAServer samples use the Sun JDBC-ODBC bridge driver that is included in the JDK versions. This driver is suitable for demonstration use only. Do not use the Sun JDBC-ODBC driver in production application deployment, or to support persistence for the EAServer message service.

Sybase Adaptive Server Enterprise and gateways

You can connect components to Sybase Adaptive Server Enterprise, OmniConnect™, or DirectConnect™ using ODBC, JDBC, or Sybase Open Client Client-Library.

To use ODBC, you must install the Sybase Adaptive Server Enterprise ODBC driver. EAServer includes ODBC drivers for some platforms. For details, see the *EAServer Installation Guide* for your platform.

To use JDBC, use EAServer Manager to define a JDBC connection cache that connects using the JDK 1.3 (or later) implementation of the jConnect driver. EAServer includes jConnect 5.5 or later runtime classes in *java/lib/jconn2.jar*, which is in the server’s CLASSPATH by default. See the jConnect documentation for details on using jConnect, including:

- The URL format for connecting to servers
- The driver class name
- Installing the jConnect metadata stored procedures on your database servers

The jConnect documentation is available on the *EAServer Technical Library* CD and is also on the Sybase Web page at http://www.sybase.com/support/manuals.

jConnect metadata support required If you have transactional components that use jConnect caches, make sure the jConnect DatabaseMetaData stored procedures are installed on the database server.

Oracle databases

You can connect to Oracle databases using JDBC, ODBC, or the Oracle Call Interface (OCI). EAServer does not include JDBC or ODBC drivers for Oracle or the OCI libraries. EAServer provides dedicated support for Oracle's proprietary C interface, OCI versions 7, 8, and 9. OCI connections that are cached by EAServer are used like any other OCI connection, except that EAServer opens the connection for you.

You can create Oracle connection caches that use ODBC or JDBC as for any other ODBC or JDBC cache, as described in "Other databases" on page 77. For information on how to use an OCI cache in a C or C++ component, see "Using OCI 7.x connection caches" in the *EAServer Programmer's Guide*.

Other databases

EAServer components can connect to any database for which an ODBC or JDBC driver is available. Contact your database vendor for information on JDBC or ODBC driver availability.

Do not use the Sun JDBC-ODBC bridge driver This driver is suitable for demonstration use only. Do not use the Sun JDBC-ODBC driver in production application deployment, or to support persistence for the EAServer message service.

To achieve ODBC connectivity, install the appropriate driver, configure a data source that uses the ODBC driver to connect to the target database, then define an EAServer connection cache that uses that data source.

To achieve JDBC connectivity, install the appropriate JDBC driver, then define an EAServer connection cache that uses that driver to connect to the target server. When you run the server, you must include the path to the driver's class file in your system's CLASSPATH setting. Many drivers require further configuration. For example, you may have to install stored procedures on each server that you intend to connect to. See your JDBC driver documentation for further configuration instructions.

Note Most JDBC drivers provide standalone test utility applications (or at least sample applications) that you can use to verify that the driver and data sources are correctly configured. Verify connectivity with a standalone application before configuring EAServer to use a new driver and data source combination.

See "Configuring connection caches" on page 78 for information on defining connection caches.

Configuring connection caches

A connection cache maintains a pool of available connections that EAServer components use to interact with third-tier data servers. You must configure connection caches for the specific user and database combinations used by your components. A connection cache entry improves performance by eliminating the overhead associated with setting up a connection when one is required. Create as many connection caches as you need.

See Chapter 26, "Using Connection Management," in the *EAServer Programmer's Guide* for additional information.

You must refresh a cache before any changes to the cache properties take effect, and you should test the connection with ping before trying to access it from components.

You cannot define two distinct caches that use identical values for server, user, password, and connectivity library. If two caches are defined with matching values for these settings, and your application requests one, EAServer returns the first match that is found.

❖ **Creating a new connection cache**

1 Highlight the Connection Caches folder.

2 Select File | New Connection Cache.

3 Configure the connection cache properties in the wizard.

Configured connection cache entries appear on the right side of the window of EAServer Manager when you highlight the Connection Cache folder on the left side of the window.

❖ **Viewing, modifying, or deleting a connection cache entry**

1 Expand the Connection Caches folder.

2 Highlight the connection cache you want to modify.

3 From the File menu, select one of:

- Configuration Wizard – a wizard to walk you through the configuration of common properties.
- Properties – displays the connection cache's properties in a single dialog box and allows you to change them.
- Delete – deletes the connection cache from the system.

General properties

The properties that you configure on the General tab are:

- **Connection Cache Name** The name for this cache configuration. Connection cache names are limited to one word, which can contain letters, numbers, and underscores. Names are case-sensitive. You cannot modify the name of an existing connection cache.
- **Description** A description of the connection cache. The description is a string with a maximum of 255 characters.
- **Database Type** The database type. Required for caches that use the EAServer automatic persistence or stateful failover features, such as EJB CMP entity beans.Several types are predefined, and you can create your own. See "Database type setting" on page 91 for more information.
- **Server Name** For each of these database driver types, enter:
  - JDBC – the URL appropriate for use in JDBC calls. For more information, see "About JDBC connection URLs" on page 80.
  - ODBC – the ODBC data source name. See "About ODBC data source names" on page 82 for more information.

- CTLIB – the server name as it would be specified in a ct_connect call. On UNIX platforms, the server must be listed in the EAServer *interfaces* file. For Windows, it must be listed in the *ini\sql.ini* file.
- OCI 7.X – the Oracle SQL*Net connect string or database alias.
- OCI 8.X – the Oracle SQL*Net connect string or database alias.
- OCI 9.X – the Oracle SQL*Net connect string or database alias.

For XA connections, enter the name of the XA resource server. For JDBC, enter:

```
NetworkProtocol=Tds:Server=yourServer:Port=TdsPort
```

where *yourServer* is the server that the driver is installed on, and *TdsPort* is the port number where TDS calls are received.

For ODBC or JDBC connections, see your driver documentation for more information. For OCI connections, see your Oracle documentation.

jConnect metadata support required If you have transactional components that use jConnect caches, make sure the jConnect DatabaseMetaData stored procedures are installed on the database server.

- **User Name** The user name for this cache. The name used (along with a password) to connect to the database identified by the server entry.

  The cache user name and password are always used for the initial connection, but you can configure set-proxy support so that a component's database work is done using the client identity. See com.sybase.jaguar.conncache.ssa on page 425 for more information.

- **Password** The password for this cache. The password and user name are used to connect to the database identified by the server entry. Passwords are encrypted in the EAServer configuration file.

  EAServer Manager does not display passwords for existing caches. To change a password, enter the new password and click OK.

About JDBC connection URLs

For JDBC connection caches, the Server Name field is a URL whose syntax depends on the JDBC interface level that the driver class supports.

JDBC level 1 drivers These driver classes implement the java.sql.Driver interface and typically have class names that end in Driver. For example, the Sybase jConnect level 1 driver class is com.sybase.jdbc2.jdbc.SybDriver, which requires syntax like:

```
jdbc:sybase:Tds:host:2638
```

The Oracle level 1 driver is class oracle.jdbc.driver.OracleDriver, which requires syntax like this example:

```
jdbc:oracle:thin:@dbserver:1521:orcl
```

For details on the jConnect syntax, see the jConnect documentation at http://sybooks.sybase.com/jc.html. For details on drivers from other vendors such as Oracle, see the vendor documentation.

JDBC level 2 connection pool drivers These driver classes implement the javax.sql.ConnectionPoolDataSource interface and typically have class names that end in PoolDataSource. For example, the Sybase jConnect level 2 driver is com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource, which requires syntax like:

```
NetworkProtocol=Tds:Server=hostname-or-IPaddress:Port=port-number
```

The Oracle level 2 driver class is oracle.jdbc.pool.OracleConnectionPoolDataSource, which requires syntax like this example:

```
ServerName=server:Port=1521:DatabaseName=ORC:DriverType=thin
```

Level 2 drivers support a *setter method* naming scheme to indicate the connection properties that they support. A setter method named set*Property* indicates that the driver supports the property named *Property*. For example, a method named setServer with a string input parameter indicates that the driver recognizes the Server property. You can set any string-valued property in the server URL. EAServer follows these rules to parse the URL:

- A colon (:) acts as a property separator.
- Items separated by an equal sign (=) are a property-value pair, of the form ***Name*=*Value*.**

You can use the Java javap command to list the property setter methods supported by your level 2 driver (or by the DataSource driver class that it extends). Driver properties that do not take string types can be set in the connection cache property com.sybase.jaguar.conncache.config-property, described in com.sybase.jaguar.conncache.config-property on page 420. Use the Advanced tab to set this property. For information on what each property configures, see the driver documentation.

About ODBC data source names

On UNIX platforms, ODBC data source names used in EAServer must be defined in the ODBC driver manager configuration, typically by modifying the *odbc.ini* file in the driver manager installation that is included with EAServer. For instructions, see the *EAServer Installation Guide* for your platform.

On Windows platforms, EAServer uses SQLDriverConnect to establish ODBC connections, which allows you to specify driver-specific connection information. You can specify either a data source and one or more optional parameters, or a DSN file that contains the data source and other parameter information.

If you use a DSN file to specify connection information, all the servers in an EAServer cluster can share the information. Use this syntax to specify a DSN file, where *file.dsn* identifies the file name:

FILEDSN=*file.dsn*

Minimally, a DSN file must contain the keyword "DRIVER" and the name of the data source in the ODBC section. The following is a sample ODBC section for an ASA 9.0 database:

```
[ODBC]
DRIVER=Adaptive Server Anywhere 9.0
UID=dba
Compress=No
DisableMultiRowFetch=No
Debug=No
Integrated=No
AutoStop=Yes
EngineName=asademo9
DatabaseFile=C:\Sybase\SQL Anywhere 9\asademo.db
```

See your ODBC driver documentation for information about which parameters can be used with the driver.

To specify a data source, instead of a DSN file, use this syntax, where *dataSource* is the name of the data source, and *param1*, *value1*, *param2*, and *value2* are optional driver-specific connection parameters and their corresponding values:

DSN=*dataSource*;[*param1=value1*];[*param2=value2*];

Note These restrictions apply to SQLDriverConnect strings:

- Neither keywords nor their values can contain the []{}(),;?*=!@\ characters.

- Data source names cannot contain a backslash or a leading blank.
- Spaces are not allowed on either side of the equal sign in a keyword/value pair.

For more information about using SQLDriverConnect, see the MSDN Web page at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odch21dpr_3.asp.

Caching properties

Configure these properties on the Caching tab:

- **Enable Cache-by-Name Access** Select this option to allow retrieval of a database connection using the connection cache name instead of requiring a user name and password.

  Cache-by-name access allows you to use caches in components without requiring access to the database password from the component source code.

  Cache-by-name access is required for JDBC caches that are used in EJB or Web application resource references.

- **Enable Connection Sanity Check** Specifies whether connections should be verified.

  Components may release a connection that is not ready for use by another component. For example, there may be unretrieved results on the connection. When the option is enabled, EAServer verifies that the connection is open and accepts commands before making it available for reuse. Disabling the option increases performance, but may complicate debugging.

  The default query for testing the connection is “select 1”. If this syntax is invalid for the database, you must configure a valid test query by setting the com.sybase.jaguar.conncache.check property on the Advanced tab. For more information, see com.sybase.jaguar.conncache.check on page 419.

- **Maximum Connections** The maximum number of connections that can be allocated before the value of Wait for Connection determines whether to wait for a connection. The value of this property should be greater than or equal to the value of Maximum Connection Pool Size. A value of 0 indicates that new connections should be opened whenever they are required. Excess connections are deallocated and not returned to the cache.
- **Wait for Connection** If the number of connections that have been allocated is equal to the value of Maximum Connections, specifies whether to wait for a connection.
- **Maximum Connection Pool Size** The maximum number of connections in the pool.

  After a connection is released, it is returned to the pool. The default value is 10. You can increase this number if performance suffers due to an insufficient number of available connections.
- **Minimum Connection Pool Size** The minimum number of connections in the pool. EAServer preallocates and opens the specified number of connections at start-up time. The default is 0.

  If no minimum size is specified, connections are opened to fill the pool up to the maximum size, on an as-needed basis.
- **Pooled Connection Idle Timeout** Specifies the number of seconds an idle connection remains in the pool before it is dropped. The default is 300 seconds (5 minutes).
- **Pooled Connection Refresh Rate** The refresh rate for the cache. The default is 600 seconds (10 minutes).

For information on tuning these settings, see “Tuning the cache size” in the *EAServer Performance and Tuning Guide*.

Driver properties

Configure these properties on the Driver tab:

- **Database Driver Type** Select the connection library type used for this cache. Your choices for library type are:
  - JDBC – for connections using a Java Database Connectivity driver.
  - CTLIB_110 – for Sybase Open Client Client-Library connections.
  - ODBC – for connections using an open database connectivity driver.

- OCI_7 – for connections using OCI 7.*x*.
- OCI_8 – for connections using OCI 8.*x*.
- OCI_9 – for connections using OCI 9.*x*.
- OCI_9U – for connections using OCI 9.*x*.

- **Connection Library 1PC** Enter the class name or library name used to support single-phase commit transactions.

  For UNIX platforms, use one of the connection libraries listed in Table 4-1.

Table 4-1: UNIX platform connection libraries

Type	Connection library name
JDBC	The Java class name for the driver class. For example, if you are using a Sybase ASA or Adaptive Server Enterprise database, set one of these values: • com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource To access the connection cache with JNDI, you must use this driver and this syntax for the server name: NetworkProtocol=Tds:Server=<*db_host_name*>:Port=<*db_listener_port*> • com.sybase.jdbc2.jdbc.SybDriver When you use this driver, use this syntax for the server name: jdbc:sybase:Tds:<*db_host_name*>:<*db_port*> The Oracle connectivity driver requires oracle.jdbc.pool.OracleConnectionPoolDataSource. For EJB CMP entity beans, EAServer supplys JDBC wrapper drivers that can improve performance through the use of deferred updates and stored procedures. For more information, see "Using CMP JDBC wrapper drivers" in the *EAServer Performance and Tuning Guide*.
Client Library 12.5	• *libjct_r.so* for Solaris, AIX, Digital UNIX, and Linux. • *libjct_r.sl* for HP-UX. You must use the version in the EAServer *lib* subdirectory, which has been optimized for EAServer threading.
ODBC	• *libodbc.so* (installed in *$JAGUAR/intersolv/lib*) for Solaris and AIX. • *libodbc.so* (installed in */var/opt/DAU100/connect/lib*) for Digital UNIX. • *libodbc.sl* (installed in *$JAGUAR/intersolv/lib*) for HP-UX.
OCI 7.x OCI 8.x OCI 9.x	• *libclntsh.so* for Solaris, AIX, Digital UNIX, and Linux. • *libclntsh.sl* for HP-UX.

Library location must be in library path On Solaris, Digital UNIX, and Linux, the location of the connection library must be in your LD_LIBRARY_PATH for ODBC, Client-Library, or OCI caches, and in CLASSPATH for JDBC caches.

On AIX, the file must be in your LIBPATH for ODBC, Client-Library, or OCI caches, and in CLASSPATH for JDBC caches.

On HP-UX, the file must be in your SHLIB_PATH for ODBC, Client-Library, or OCI caches, and in CLASSPATH for JDBC caches.

For Windows platforms, use one of the connection libraries listed in Table 4-2.

Table 4-2: Windows platform connection libraries

Type	Connection library name
JDBC	The Java class name for the driver class. For example, if you are using a Sybase ASA or Adaptive Server Enterprise database, set one of these values: • com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource To access the connection cache with JNDI, you must use this driver and this syntax for the server name: NetworkProtocol=Tds:Server=<*db_host_name*>:Port=<*db_listener_port*> • com.sybase.jdbc2.jdbc.SybDriver When you use this driver, use this syntax for the server name: jdbc:sybase:Tds:<*db_host_name*>:<*db_port*> The Oracle connectivity driver requires oracle.jdbc.pool.OracleConnectionPoolDataSource
Client Library 12.5	*libjct.dll* – you must use the version in the EAServer *dll* subdirectory, which has been optimized for EAServer threading.
ODBC	*odbc32.dll*
OCI 7.*x*	*ociw32.dll*
OCI 8.*x*	*oci.dll*
OCI 9.*x*	*oci.dll*

Library location must be in PATH or CLASSPATH The location of the connection library must be in your PATH environment variable for ODBC, Client-Library, or OCI caches, and in CLASSPATH for JDBC caches.

- **Connection Library XA** Enter the class name or library name used to support two-phase commit transactions, and the name of the XA resource library.

Table 4-3: Connection libraries and XA resource libraries for Solaris

Database driver	Connection library	XA resource library
JDBC	Same as the JDBC connection library described in Table 4-1 on page 85	com.sybase.jdbc2.jdbc.SybXADataSource
Sybase Client Library 11.0	*libjct_r.so*	*libjxa.so*
Oracle OCI 7.x	*libclntsh.so*	*libclntsh.so*
Oracle OCI 8.x	*libclntsh.so*	*libclntsh.so*
Oracle OCI 9.x	*libclntsh.so*	*libclntsh.so*

Table 4-4: Connection libraries and XA resource libraries for Windows

Database driver	Connection library	XA resource library
JDBC	Same as the JDBC connection library described in Table 4-2 on page 86	com.sybase.jdbc2.jdbc.SybXADataSource
Sybase Client Library 11.0	*libjct.dll*	*libjxa.dll*
Oracle OCI 7.x	*ociw32.dll*	*xa73.dll*
Oracle OCI 8.x	*oci.dll*	*oraclient8.dll*
Oracle OCI 9.x	*oci.dll*	*oraclient9.dll*

By default, EAServer uses the XA libraries listed in Table 4-3 and Table 4-4 to obtain an XA resource that is exported from the database connection libraries.

The resource manager determines the connection type based on the configuration and state of the transaction. In most cases, you should not need to modify the resource manager properties. However, to use a shared library or DLL other than the default, you must edit the database property file. For example, for XA resource connections using Oracle OCI 8.*x*, where 8.*x* is 8.1.5 or lower, set this property value in the *OCI_8.props* file:

```
com.sybase.jaguar.resourcemanager.xalib = xa80.dll
```

Table 4-5 lists the resource manager database property files for the C/C++ connection caches, which are located in the EAServer *Repository/ResourceManager* directory.

Table 4-5: Database property files

Database driver	Database property file
Sybase Client Library 11.0	*CTLIB_110.props*
Oracle OCI 7.x	*OCI_7.props*
Oracle OCI 8.x	*OCI_8.props*
Oracle OCI 9.x	*OCI_9.props*

To use a shared library or DLL other than the default for a single connection cache, edit the connection cache properties file, *<cache_name>.props*, located in the EAServer *Repository/ConnCache* directory, and specify the shared library or DLL name:

```
com.sybase.jaguar.conncache.xadllname = library_name
```

If you execute a transaction without an XA resource configured for a database, the EAServer connection manager returns CS_FAIL.

If a configured XA resource is not running or cannot be connected to, EAServer cannot initialize. To solve this problem, perform one of these tasks:

- Start the XA resource.
- Make the XA resource available to EAServer.
- Start the JagRepair server and delete the XA resource. See "Using the JagRepair server" on page 55 for information about JagRepair.

- **Use HA Connection** When using Client Library 11.0 to connect to an Adaptive Server Enterprise database, this enables the high availability failover feature.

  To use this feature, you also need to modify the client connection information and pass CS_HAFAILOVER when calling ct_config or ct_con_props—see "Open Client Client-Library" on page 158 for more information.

XA properties

Configure these properties on the XA tab:

- **Database Name** If you select OCI 7.x, 8.x, or 9.x, specify the database name. Other connection cache types do not require a value.
- **Default Open String** The string used to connect to the XA resource. You cannot modify this string, which is automatically built from the information that you entered in the other tabs.
- **Open String Suffix** In this optional field, you can specify any valid open string options. For example, for a Sybase Client-Library 11.0 XA resource, you can specify a log file name by entering:

  -L *logfile*

If you specify a log file, any errors related to Open Client XA operations are written to this file. If you do not specify a log file, or set the open string suffix incorrectly, errors are logged to the *xa_syb_<pid>.trc* file, where *pid* is the EAServer process ID.

Note If the Open String Suffix is set incorrectly, XA operations are not supported for the connection cache.

- **Close String** In this optional field, you can specify a value used by the resource to close a connection.

SQL tracing properties

Configure these properties on the SQL Tracing tab:

- **Enable SQL Tracing** If you select this option, all SQL connections are traced. Trace statements, which include the method name, input parameters, and output values, are sent to the device specified in Logging Handler.
- **Log by Connection** Select this option to create separate log files for each connection. Log file names are created using the cache name and an integer to represent the connection; for example, if the cache name is JavaCache, the log files are *JavaCache1.log*, *JavaCache2.log*, and so on.
- **Log Parent Handlers** If you select this option, trace statements are sent to parent handlers, in addition to the device specified in Logging Handler. If the Logging Handler and its parent handlers point to the same device, trace statements can be duplicated.
- **Logging Handler** Select the device to handle trace statements:
  - eas_servlet – directs trace statements to the servlet logging handler. To configure where the output is sent, see “Configuring log profiles” on page 56.
  - eas – writes trace statements to the server log file; for example, *Jaguar.log*.
  - eas_console – writes trace statements to the console.

Advanced tab

To set the cache properties described in this section, use the Advanced tab. You can set any property on this tab, as listed in Connection cache properties on page 418. In most cases, the controls on the other tabs provide a more convenient way to set properties.

You can also configure additional driver- or library-specific properties on this tab.

JDBC driver properties

For a JDBC connection cache, you can specify settings beyond those shown in the Connection Cache Properties dialog using the com.sybase.jaguar.conncache.config-property property. For details, see com.sybase.jaguar.conncache.config-property on page 420.

For backward compatibility, cache properties that are defined in this form are still supported:

```
propertyName=value
```

Any property whose name does not begin with com.sybase.jaguar is passed to the JDBC driver as a connection property. For example:

```
PACKETSIZE=2048
```

Different JDBC drivers recognize different sets of properties. See your driver documentation for more information on the properties that can be set.

Client-Library connection settings

You can set the following properties for Client-Library connections. See the Client-Library documentation for the ct_con_props routine for more information:

- CS_HOSTNAME
- CS_APPNAME
- CS_TDS_VERSION
- CS_PACKETSIZE

This example sets all of these properties:

```
CS_HOSTNAME=myhostname
CS_TDS_VERSION=CS_TDS_46
CS_PACKETSIZE=512
```

```
CS_APPNAME=myapp
```

Any property name not recognized by EAServer is ignored.

The CS_TDS_VERSION property takes one of these values: CS_TDS_50, CS_TDS_495, CS_TDS_46, CS_TDS_42, or CS_TDS_40.

The CS_PACKETSIZE property takes a numeric value within the range of legal packet sizes for the server. If the server does not support the packet size, the cache cannot connect to the server.

Note Make sure there is no extra white space between the property name, the equal sign, and the property value, or after the property value.

Database type setting

In connection caches used to support automatic persistence or stateful failover, you must configure the Database Type connection cache property. This property defines database-specific information required by the storage component, for example, the commands to verify a table exists and create new tables. Several types are predefined, as described in Table 4-6.

Table 4-6: Database type identifiers

Database identifier	To indicate
Oracle8i	Oracle version 8
Oracle9i	Oracle version 9
SQL_Server	Microsoft SQL Server
Sybase_ASA	Sybase Adaptive Server Anywhere
Sybase_ASE	Sybase Adaptive Server Enterprise
Unknown	Any other database

You can create your own database type definitions as described in Database type properties on page 432.

Connection cache refresh

If you have just modified connection cache properties, refresh the cache before you test it.

To refresh the cache:

1 Highlight the Connection Caches folder.

2 Highlight the cache's icon and choose File | Refresh.

Refreshing a cache may affect running components that are using the cache, specifically:

- If you change the connectivity library setting, cache references held by components become invalid. Attempts to retrieve connections or query cache properties will cause errors. In this case, the component must retrieve a new cache handle.
- If you change other properties, such as user name, password, server name, or the number of connections in a cache, cache references remain valid, but components may be affected by the changed settings. For example, if you change the server name, connections retrieved after the cache has been refreshed will go to the server indicated by the new name.

Connection cache ping

This feature allows you to test the cache configuration to verify that connections can be made using the supplied parameters. If you have just changed any settings, refresh the cache before testing it.

To test the cache with ping:

1 Highlight the Connection Caches folder.

2 Highlight the cache's icon and choose File | Ping.

3 EAServer Manager reports whether the connection attempt succeeded.

If ping fails, check the message text for a description of the problem. The server log file may contain additional information about the cause of the error.

If you change the cache properties to correct the problem, you must refresh the cache before testing again.

Using XA resources with Adaptive Server Enterprise

To use XA resources with an Adaptive Server Enterprise 12.0 or later database running on UNIX or Windows, perform these tasks:

1. (Windows only.) Verify that you have the correct license file, which is located in *%SYBASE%\SYSAM-1_0\licenses\license.dat*. To run XA transactions, you must have the ASE_DTM license.
2. Configure the database to use XA transactions. For more information, see *Using the Adaptive Server Distributed Transaction Management Features*, available on the Sybase Product Manuals Web page at http://manuals.sybase.com/onlinebooks/group-as/asg1250e.
3. Create a database to use for XA transactions.
4. Configure the database tables and stored procedures required for JDBC connection caches, which are available in *$JAGUAR/html/classes/sp/sql_server12.sql*.
5. Configure the database tables required for C/C++ connection caches. You can install these tables by running *xa_load.sh* (UNIX) or *xa_load.bat* (Windows), which is located in the EAServer *html/classes/sp* directory.
6. Create a login, other than sa, for connecting to the database. You cannot use the sa login to connect to an Adaptive Server Enterprise databases for XA transactions.

Modifying the Adaptive Server configuration files

When you configure an XA resource for C/C++ connection caches, you need to modify two configuration files.

On UNIX:

- *$JAGUAR/interfaces*
- *$JAGUAR/xa_config*

On Windows:

- *%JAGUAR%\ini\sql.ini*
- *%JAGUAR%\ini\xa_config*

interfaces and *sql.ini*

On UNIX, add these lines to the *interfaces* file:

```
DbServer
   master tcp ether host 5300
   query tcp ether host 5300
```

On Windows, add these lines to the *sql.ini* file:

```
[DbServer]
master=NLWNSCK,host,5200
```

```
query=NLWNSCK,host,5200
```

where *DbServer* is the name of the database server and *host* is the name of the machine on which the database server runs. *DbServer* should have the same value that you entered in the Server Name field when you configured the XA resource.

xa_config

For both operating systems, add these lines to the *xa_config* file:

```
[xa]
   lrm=serverName
   server=DbServer
```

where:

- *serverName* is the name you entered in the Server Name field when you configured the XA resource.
- *DbServer* is the name of the server you entered in either the *interfaces* file or the *sql.ini* file.

Note *serverName* and *DbServer* represent the same server and should have the same value.

Configuring connectors

A connector is a specialized connection factory that provides connections for EJBs, Java servlets, JSPs, and CORBA-Java components.

Each connector has one managed connection factory with its own property file. The Java Connection Manager (JCM) classes create the connection factories and manage a pool of connections for a connector.

Transaction modes

EAServer connectors support these transaction modes:

Transaction attribute	Description
NO_TRANSACTION	The connector is nontransactional; it does not support local transactions or XA resources.
LOCAL_TRANSACTION	The connector implements only the LocalTransaction interface, therefore, EAServer must manage all transactions.
XA_TRANSACTION	The connector supports both local transactions and XA transactions.

For information on importing and exporting connectors, see “Deploying connectors” on page 195.

❖ **Configuring a connector**

1 Expand the Connectors folder, then highlight the connector you want to configure.

2 Choose File | Properties.

3 The Connector Properties dialog box has four tabs.

a On the General tab:

1 Enter a description for the connector.

2 Set the Configured Queue Size; the default size is 10. This defines the maximum number of connections that can be pooled. When a component requests a connection, EAServer attempts to get it from the connection pool. If none exists, it opens a new connection.

3 Set the Idle Connection Timeout, in seconds. Connections are dropped from the pool when they have been idle for this amount of time. The default is 0, which means the connection never times out.

b On the Config Properties tab, add environment entries:

1 Click Add. This adds a new row to the Environment Entries Referenced in Code list.

2 In the Entry column, enter `ConnectionURL`.

3 In the Type column, choose String.

4 In the Value column, enter the JNDI name for the resource; for example, `jdbc:sybase:Tds:myhost:2638`.

5 In the Description field, enter a brief description of the connector.

c On the Java Classes tab:

1 Click Add. This adds a new row to the Java Classes list.

2 Enter the name of the JAR file that implements the connector.

You can repeat these steps to add other JAR files required to run the connector. These JAR files are loaded by the EAServer custom class loader and reloaded when you refresh the connector. Classes or JAR files not listed here are loaded by the Java system class loader, which means you must restart the server to use new versions of the classes.

d The Advanced tab lists all the properties associated with the connector.

Note You cannot test the connections obtained from a connector using ping.

❖ **Adding a managed connection factory**

For each connector, add one managed connection factory.

1 Expand the Connectors folder, then highlight the connector to which you want to add a managed connection factory.

2 Choose File | New Managed Connection Factory.

3 Choose File | New Managed Connection Factory Wizard. Follow the wizard pages to configure the new connector. If you require more information on any setting, click Help in the wizard.

❖ **Configuring connector properties**

1 Highlight the icon for the managed connector factory that you want to configure. Choose File | Properties.

2 The Connection Factory Properties dialog box, which has four tabs, displays:

a On the General tab:

1 Enter a description for the managed connection factory.

2 Set the Configured Queue Size.

3 Set the Idle Connection Timeout, in seconds.

b On the Config Properties tab, add environment entries.

1 Click Add. This adds a new row to the Environment Entries Referenced in Code list.

2 In the Entry column, enter “ConnectionURL”.

3 In the Type column, choose String.

4 In the Value column, enter the JNDI name for the resource; for example, `jdbc:sybase:Tds:myhost:2638`.

5 In the Description field, enter a brief description of the managed connection factory.

c On the Security Properties tab, enter a user name and password that are valid to access the resource.

d The Advanced tab lists all the properties associated with the managed connection factory.

Within an application, you can use JNDI to look up a connector's managed connection factory instance and get a connection to an enterprise information system, as this code sample illustrates:

```
// Get the initial JNDI context
Context initContext = new InitialContext();

// Look up a connection factory instance
javax.resource.cci.ConnectionFactory cf =
   (javax.resource.cci.ConnectionFactory)
   initCtxt.lookup("java:comp/env/eis/MyEIS);

javax.resource.ci.Connection conn = cf.getConnection();
```

❖ **Synchronizing a connector**

You can synchronize a connector within a cluster of servers by using EAServer Manager to copy standalone connectors and their property files to other servers.

1 Expand the Connectors folder, then highlight the connector you want to synchronize.

2 Choose File | Synchronize. This displays the Synchronize Connector dialog box.

3 See Table 6-1 on page 136 for a description of the synchronization properties.

4 Click Start Sync. The Synchronization message box displays the status of the process.

❖ **Refreshing a connector**

1 Expand the Connectors folder, then highlight the connector you want to refresh.

2 Choose File | Refresh.

❖ **Refreshing a connector view**

1 Expand the Connectors folder, then highlight the connector whose view you want to refresh.

2 Choose File | Refresh View.

❖ **Deleting a connector**

1 Expand the Connectors folder, then highlight the connector you want to delete.

2 Choose File | Delete Connector.

For more information on managing connections, see Chapter 26, "Using Connection Management," in the *EAServer Programmer's Guide*.

Configuring Adaptive Server Enterprise connections

When EAServer uses an Adaptive Server Enterprise database running on UNIX, you can configure the number of user connections, but it cannot exceed the number of file descriptors available to Adaptive Server on the operating system. When configuring Adaptive Server user connections, the system administrator should consider the number of file descriptors available for each process. Although most of the open file descriptors are available for user connections, a few are used by Adaptive Server for opening files and devices.

When EAServer uses an Adaptive Server Enterprise database running on Windows, you can create more than 6000 user connections.

For additional information on user connections, see the *Adaptive Server Enterprise System Administration Guide* on the Sybase Product Manuals Web site at http://manuals.sybase.com/onlinebooks/group-as/asg1250e.

For Sun Solaris and SGI

For Sun Solaris, you can set both *soft* and *hard* limits for file descriptors. The soft limit can be increased up to the hard limit by the user, but the hard limit can be increased only by someone with "root" permissions. The soft limit determines the number of open file descriptors available to an Adaptive Server engine.

Displaying current soft and hard limits

To display the current soft limit, for C shells, enter:

```
limit descriptors
```

For Bourne shells, enter:

```
ulimit -n
```

To display the current hard limit for C shells, enter:

```
limit -h descriptors
```

For Bourne shells, enter:

```
ulimit -Hn
```

Increasing the soft limit

To increase the soft limit for C shells, enter:

```
limit descriptors n
```

For Bourne shells, enter:

```
ulimit -Sn new_value
```

where *n* is the current value for the soft limit, and *new_value* is the value to which you want to increase the soft limit.

Note You can use the preceding commands in your *runserver* file to increase the hard and soft limits. Because the *runserver* file is a Bourne shell script, be sure to use the Bourne shell versions of these commands in the *runserver* file.

Increasing the hard limit

❖ **Setting up the sample program to increase the hard limit**

1 Use an ASCII editor to create *file_name.c* (where *file_name* is the name you give the file). Type the text shown in the sample in "Sample program" on page 101.

2 Compile the file:

```
cc file_name.c -o program_name
```

where *file_name* is the name of the source file you created, and *program_name* is the name you want to give the program.

3 Change the program's permissions and ownership so that it will execute as "root":

```
chmod 755 program_name
chown root program_name
```

where *program_name* is the name of the compiled program.

4 The "root" user can run the program to start Adaptive Server with increased user connections by typing this at the operating system prompt:

```
program_name dataserver -d master_device_name
```

where *program_name* is the name of the compiled program, and *master_device_name* is the full path of Adaptive Server's master device. Instead of typing the command at the operating system prompt, you can add *program_name* before the data server command line in the Adaptive Server *runserver* file.

For Compaq Tru64

The number of file descriptors per process is determined by the operating system parameter *open_max*. The default value of *open_max* is 4096. For more information on setting this parameter, see the Compaq Tru64 operating system documentation.

To obtain the current value of the *open_max* parameter, use the Korn or Bourne shell ulimit command:

```
ulimit -n
```

Adaptive Server can use a maximum of 1024 file descriptors, regardless of the value of *open_max*.

Use the sysconf or getdtablesize C library functions to obtain the number of current file descriptors.

For HP-UX

The kernel parameters *maxfiles* and *maxfiles_lim* control the number of file descriptors available to any one process.

Sample program

The following example shows source code you can use to increase the hard limit:

```
#include <sys/time.h>
#include <sys/resource.h>
#include <sys/types.h>
/*
** define MAX_CONNECTIONS to a number less than
** 10000. The number defined will then become the
** maximum number of connections allowed by an
** Adaptive Server.
*/
 #define MAX_CONNECTIONS 9999
 extern int errno;

 main(argc,argv)
 char **argv;
 {
     struct rlimit rlp;
     uid_t uid;

    rlp.rlim_cur = MAX_CONNECTIONS;
    rlp.rlim_max = MAX_CONNECTIONS;

  /* set the number of open file desriptors to
  ** MAX_CONNECTIONS */
    if (setrlimit (RLIMIT_NOFILE,&rlp) == -1)
    {
       perror("setrlimit");
       exit(1);
    }

   /* reset the user id to disable superuser
   ** privileges */
    uid = getuid();
    setuid(uid);

   /* run the program */
    execv(*++argv, argv);
}
```

CHAPTER 5

Naming Services

A *naming service* lets you associate a logical name with an object, such as a package and component. Naming helps EAServer applications easily locate an object anywhere on a network, then implement the referenced object.

The naming service “binds” a name to an object. The combination of bound name and its referenced object is the *name context*. The referenced object in a name context can be a component within a package or even an existing name context, the same way a named directory can contain a file or other named directory.

The collection of name context information—each object and its bound name—comprises the *namespace*. When client applications reference an object, they look to the namespace to cross-reference or *resolve* the name with the referenced object.

How does the EAServer naming service work?

The process of binding objects is performed by a name server. Each instance of EAServer can be its own name server, or you can configure a server to use another server as its name server. You can also use an external naming service, such as an LDAP server, in conjunction with EAServer’s naming service.

You set the naming service options for each server using the Naming Service tab on the Server Properties window.

EAServer initial context

The naming service relies on an "initial" or default name context for each server. You set the initial context when you set up the server's Naming Service properties.

The server name context syntax follows a specific organization or schema. You can use this schema to represent the hierarchy of objects in the namespace, for example by geographic region, organizational unit, and so on.

If you use EAServer as the name server, and do not use an external naming service, the initial context for your server uses this format:

<Level 1>/<Level 2>/<Level 3>/...

The number of levels depends on the hierarchy you want to represent. For example:

US/sybase/finance
US/sybase/marketing
US/sybase/sales

If you use an LDAP server as an external naming service, the initial context must follow the syntax and schema of the LDAP server. LDAP servers have predefined schema for common objects such as country, organization, and organizational unit. EAServer uses the following format for an LDAP-compatible initial context:

ou=<organizational unit>, o=<organization>, c=<country>

Using the previous examples, the initial contexts would be:

ou=finance,o=sybase,c=US
ou=marketing,o=sybase,c=US
ou=sales,o=sybase,c=US

On start-up, the name server binds all object implementations on EAServer to the initial context of the server on which the object is installed. Once the server binds an object, the structure of the resulting name context is:

<initial context>/<package>/<component>

where

<initial context> is the initial context property for the server where the component is installed.

<package> is the name of the package being bound, as displayed in EAServer Manager.

<component> is the name of the component being bound, as displayed in EAServer Manager.

If you have multiple EAServer installations, and one of them is designated as the name server, the name server binds the objects on those servers using the initial contexts of their respective servers. If you do not specify an initial context for any of those servers, the name server binds the objects with the initial context of the designated name server.

Note You can set the server properties to enable password protection for name binding on an EAServer name server. See "Name binding password security" on page 119.

Name binding example

To illustrate how an EAServer name server uses the initial context to create name contexts for objects on multiple servers, let's say you have three EAServer installations:

- Server A contains package `Pkg1` and components `CompX` and `CompY`. You assign the server an initial context of */us/sybase/serverA*.
- Server B contains package `Pkg2` and the component `CompZ`. You assign the server an initial context of */us/sybase/serverB*.
- Designate server C to be the name server for servers A and B by specifying the URL for server C (*iiop://myhost:9050*) in their Naming Services properties.

When you start server A, it connects to server C, using the name server URL you entered in server A's Naming Service properties. The name server gets the initial context for server A and binds each object installed on server A. The resulting name contexts are based on server A's initial context, the package name, and the components in the package. For this example, the name server creates the following bindings:

/us/sybase/serverA/Pkg1/CompX
/us/sybase/serverA/Pkg1/CompY

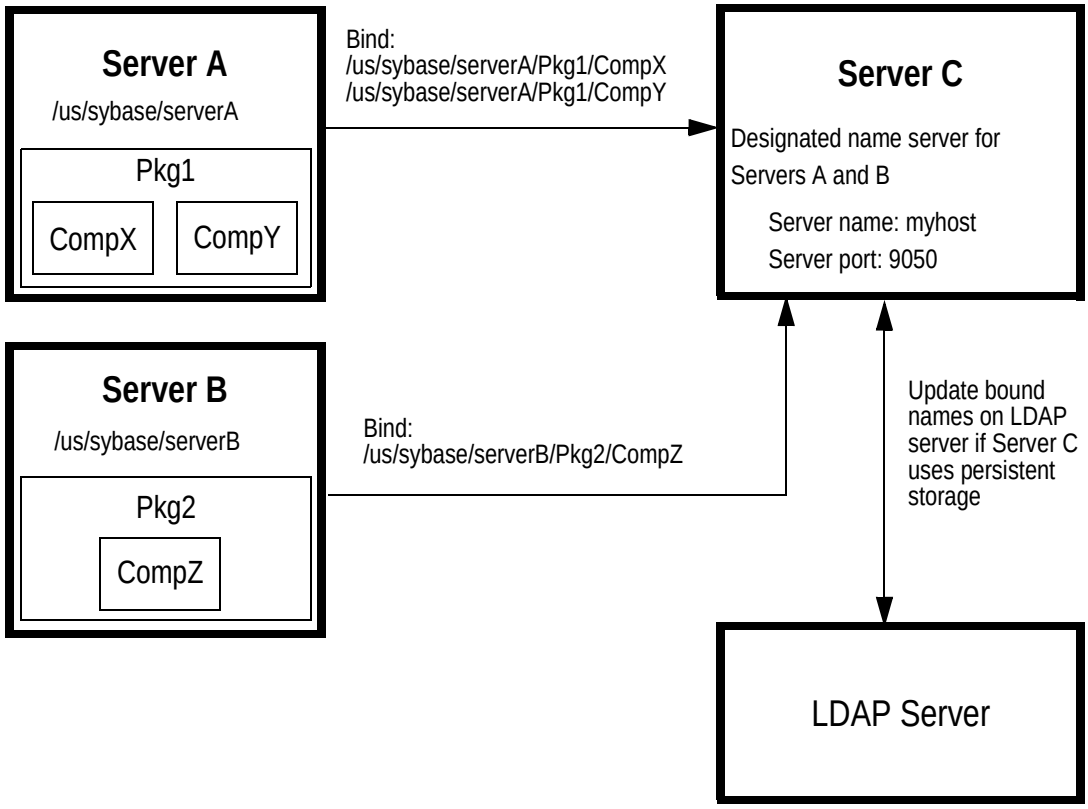
If you are using an external naming service such as LDAP, the name server also updates the existing object references on that server, if any.

When you start server B, the name server creates the following binding:

/us/sybase/serverB/Pkg2/CompZ.

Figure 5-1 illustrates the name binding process.

Figure 5-1: Name binding process



An application referencing object CompY uses the URL of the name server, followed by the object's name context. For example:

iiop://myhost:9050/us/sybase/serverA/Pkg1/CompY

The name server finds the name context in the namespace, resolves the name context with the object it references, then implements the object.

If you had not assigned an initial context to Server A, the name server, server C, would create name contexts for objects Pkg1/CompX and Pkg1/CompY using the initial context of the name server. In this case, the client application can simply retrieve CompY using this URL:

iiop://myhost:9050/Pkg1/CompY

Transient versus persistent storage

The naming service inherently provides *transient* object name storage. The name server is instantiated when you start EAServer, and binds names to all the known object references. The name server provides the bound name and object references to EAServer's session manager object. Because this information is stored in memory, the name context information is retained only as long as the server is running.

You can add *persistent* object name storage capabilities to EAServer by using an external directory naming service, such as an LDAP server. The external server retains object name information, and the EAServer name server updates this information whenever it creates new bindings or unbinds existing ones.

To use an external naming service, specify the URL of the external server in the Naming Service properties of the designated EAServer name server. You must also provide a manager DN (distinguished name) and password that has exclusive access to all objects in the LDAP server database for EAServer to be able to update the stored name context information.

CORBA CosNaming API support

The EAServer naming service is an implementation of the CORBA CosNaming component. The CosNaming component is a collection of interfaces that defines the naming service. These interfaces provide support for object binding and lookup.

EAServer implements the NamingContext interface to bind a name to an object, thereby creating a NamingContext object. Client applications use the NamingContext interface to "resolve" a bound name to its referenced object.

CosNaming::Name represents a name context that can be bound to an object implementation or another name context. CosNaming::Name is a sequence of one or more NameComponent structures. The NameComponent consists of two attributes: the identifier and the type. Both of these attributes are represented as IDL strings.

The IDL specification for NameComponent and the NamingContext interface is:

```
module CosNaming
    typedef string Istring;
    struct NameComponent {
        Istring id;
        Istring kind;
    };
    typedef sequence<NameComponent> Name;
};
```

Binding names

There are four methods to create bindings:

- bind creates a binding for a name and an object to create a name context. Name contexts created using bind are not included when compound names are passed for resolution.
- rebind creates a binding for a name and an object, even if the name is already bound to an object. Name contexts created using rebind are not included when compound names are passed for resolution.
- bind_context binds a name to an existing name context. Name contexts created using bind_context are included when compound names are passed for resolution.
- rebind_context binds a name to an existing name context, even if the name is already bound to an object. Name contexts created using rebind_context are included when compound names are passed for resolution.

To remove an object reference from a name context, EAServer uses the unbind function. When you shut down the server, all bound objects are automatically unbound using this function. However, you can also use unbind when you delete a package or component from the repository. If you use persistent name storage, use unbind to remove references to deleted packages and components on the external server.

Resolving EAServer objects

EAServer uses the resolve method to retrieve an object based on the name context into which it is bound. The name context used to retrieve an object must be identical to the object's bound name context. The naming service performs the "narrowing" of the object to the appropriate return type. In other words, the client does not need to cast the returned object to a more specialized interface.

There are two ways for Java clients to access the naming service to resolve object names:

- Using EAServer's CosNaming Java interface
- Using JNDI

Resolving objects using the CosNaming interface

The service provider interface (SPI) uses the CosNaming interface to connect to the EAServer name server and retrieve the CORBA Interoperable Object Reference (IOR) associated with the server's manager object. Once the IOR is retrieved, the naming service creates a session with the manager object and then creates an instance of the requested object. The SPI returns an instance of the requested object to the client.

After initializing the ORB, call the orb.resolve_initial_references method to obtain the initial naming context. The naming context is an object that implements the CosNaming::NamingContext IDL interface; it is used to resolve EAServer component and service names to server-side objects.

The initial NamingContext has the name context that was specified in the com.sybase.CORBA.NameServiceURL ORB initialization property. Your client program invokes the NamingContext::resolve operation to obtain an instance of the EAServer authentication service as well as component instances.

The NamingContext::resolve operation takes a CosNaming::Name parameter, which is a sequence of CosNaming::NameComponent structures. The Java definitions of these types and the NamingContext::resolve operation follow:

```
package org.omg.CORBA.CosNaming;

class NameComponent {
  public String id;    // Represents a node in a name
  public String kind;  // Unused, can contain comments
```

```
    // Construct a NameComponent instance with the
    // specified initial values for id and kind fields
    public NameComponent(String id, String kind);
}

interface NamingContext {
    ... other methods not shown ...
    public org.omg.CORBA.Object resolve
     (NameComponent[] n)
     throws
     org.omg.CosNaming.NamingContextPackage.NotFound,
    org.omg.CosNaming.NamingContextPackage.CannotProced,
    org.omg.CosNaming.NamingContextPackage.InvalidName;
}
```

In Java, a name is represented by an array of NameComponent instances, with the *id* field of each instance set to a node of the name. For example, the name:

```
USA/Sybase/Jaguar/TestPackage/TestComponent
```

can be represented by the array *theName* which is created in this code fragment:

```
import org.omg.CORBA.CosNaming.*;
import org.omg.CORBA.CosNaming.NamingContextPackage.*;
public class myApplet extends Applet {

  NamingContext nc;
  ... deleted code that retrieves initial NamingContext
...

  NameComponent theName[] = {
    new NameComponent("USA", ""),
    new NameComponent("Sybase", ""),
    new NameComponent("Jaguar", ""),
    new NameComponent("TestPackage", ""),
    new NameComponent("TestComponent", "")
  } ;
```

For convenience, the naming service allows you to specify multiple nodes of a name in one NameComponent instance, using a forward slash (/) to separate nodes. The name from the example above can be represented in a one-element array as shown below:

```
  NameComponent theName[] = {
    new NameComponent(
     "USA/Sybase/Jaguar/TestPackage/TestComponent","")
  };
```

NamingContext::resolve resolves a name to an object; this method either returns an org.omg.CORBA.Object instance or throws an exception.

For complete information about instantiating and resolving objects with CORBA naming services, see Chapter 12, "Creating CORBA Java Clients," in the *EAServer Programmer's Guide*.

Interoperable naming

EAServer supports interoperable naming for EJB 2.0, and implements the NamingContextExt interface, which enables you to look up objects with URLs and *stringified* names (an object reference converted to a string.)

The IDL specification for the NamingContextExt interface is:

```
module CosNaming {
//...
  interface NamingContextExt: NamingContext{
    typedef string URLString;
    typedef string Address;
    typedef string StringName;

    StringName to_string(in Name n) raises(InvalidName);
    Name to_name(in StringName sn) raises(InvalidName);

    exception InvalidAddress{};

    URLString to_url(in Address addrkey, in StringName sn)
                     raises(InvalidAddress, InvalidName);
    Object resolve_str(in StringName n)
                     raises(NotFound, CannotProceed, InvalidName);
  };
};
```

The com.sybase.CORBA.ORB implementation supports corbaloc and corbaname URLs, which allow you to define object references that are more readable and easier to use than IORs. The corbaloc URL format is similar to FTP or HTTP URLs and can be modified more easily than IORs. corbaloc URLs can be used for objects that are accessible through IIOP or resolve_initial_references (rir:). This is an example of the corbaloc format that uses the rir protocol, where "NamingService" is the key string that is passed to resolve_initial_references:

```
corbaloc:rir:/NamingService
```

When you use rir, you cannot use any other protocol. This example uses IIOP to look up the key string “Prod/TradingServices” on the host “555xyz.com”:

```
corbaloc:iiop1.1@555xyz.com/Prod/TradingServices
```

When you use the corbaloc format to reference beans on another server, you must be sure that both servers use interoperable security; for more information, see the *EAServer Security Administration and Programming Guide*.

A corbaname URL is similar to a corbaloc URL, but it also contains a stringified name that identifies a naming context binding. In this example, the host is “555xyz.com”, the key string is “dev/NContext1”, and “#” marks the beginning of the stringified name:

```
corbaname::555xyz.com/dev/NContext1#a/b/c
```

When an object reference does not contain an object key, the default key “NamingService” is used. In this example, “NamingService” is used to look up a NamingContext, then the stringified name “a/b/c” is used to resolve the final object:

```
corbaname::555.xyz.com#a/b/c
```

You can reference local beans (those that run within the same Java VM by prefixing the bean name with “local:”; for instance, `local:ejb/MyBean`.

To define this reference using the corbaname format, use this syntax:

```
corbaname:rir:#ejb/MyBean
```

For information on using interoperable naming URLs for EJB 2.0 components and clients, see Chapter 9, “EAServer EJB Interoperability,” in the *EAServer Programmer’s Guide*.

Network name service

The Network repository type provides a mapping between a client’s location and the name servers that respond to the client’s request. A mapping specifies either an IP address to a location name, or a location name to a name server. Here are some sample network properties:

```
203.97.*.* = Wellington
Wellington=iiop://ns-wgtn.sybase.co.nz:9000,iiop://ns-west.sybase.com:9000
Emeryville=iiop://ns-west.sybase.com:9000,iiop://ns-east.sybase.com:9000
```

JNDI support

Java Naming and Directory Interface (JNDI) is a standard Java interface for accessing distributed objects and services by name. It provides a portable, unified interface for naming and directory services. The JNDI specification is independent of any specific directory or naming service such as LDAP, NDS, DCE/CDS, or NIS.

EAServer's JNDI implementation includes the JNDI service provider interface (SPI), which enables you to use a variety of custom directory and naming services. EAServer uses the SPI in conjunction with the CosNaming interface to provide component lookup capability. Given a bound name, the SPI locates the referenced package and component. Once it locates the component, the SPI works with the client stub interface to instantiate the component and return the requested object.

For complete information about instantiating and resolving objects with JNDI, see Chapter 12, "Creating CORBA Java Clients," in the *EAServer Programmer's Guide*.

Note When you start the server, the JNDI classes required for the server's JDK version are configured automatically.

JNDI J2EE features

In J2EE, you can use the application component's naming environment to customize an application's business logic without accessing the source code. The application component's container implements the environment as a JNDI naming context and provides the JNDI interfaces to access the environment properties that you define in the deployment descriptor.

Environment properties

When you deploy a J2EE application, use the deployment descriptor to define all the environment properties that the application component needs to access. This sample code defines the environment property (env-entry) *maxExemptions* as an Integer and sets its value to 10:

```
<env-entry>
   <description>
     The maximum number of tax exemptions
   </description>
```

```
    <env-entry-name>maxExemptions</env-entry-name>
    <env-entry-type>java.lang.Integer</env-entry-type>
    <env-entry-value>10</env-entry-value>
</env-entry>
```

The information between the opening and closing env-entry tags defines an environment entry element, which consists of:

- **description** This is optional.
- **env-entry-name** The environment property name, relative to the *java:comp/env* context.
- **env-entry-type** The environment property's Java datatype must be one of: Boolean, Byte, Double, Float, Integer, Long, Short, or String.
- **env-entry-value** The environment property value, which is optional.

Within the same container, all instances of an application component share the same environment properties. The component instances cannot modify the environment at runtime.

An application component instance uses the JNDI interfaces to locate the environment naming context and access the environment properties. To locate the naming context, an application creates a javax.naming.InitialContext object and gets the InitialContext for *java:comp/env*. In this example, the application retrieves the value of the environment property *maxExemptions* and uses that value to determine an outcome:

```
Context initContext = new InitialContext();
Context myEnv =
    (Context)initContext.lookup("java:comp/env");

// Get the maximum number of tax exemptions
Integer max=(Integer)myEnv.lookup("maxExemptions");

// Get the minimum number of tax exemptions
Integer min = (Integer)myEnv.lookup("minExemptions");

// Use these properties to customize the business logic
if (numberOfExemptions > max.intValue() ||
    (numberOfExemptions < min.intValue())
     throw new InvalidNumberOfExemptionsException();
```

Default name service When you call the empty constructor to create a new InitialContext, EAServer sets the Context.INITIAL_CONTEXT_FACTORY system property and sets EAServer's EJB name service as the default.

For information about using EAServer Manager to add and configure environment properties for a Web application, application client, or EJB component, see "Environment properties" in Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide*.

EJB references

An EJB reference identifies the home of an enterprise bean. You can use the deployment descriptor to create a link between an EJB reference and an enterprise bean, contained within an EJB-JAR file. Deployment descriptor interfaces allow an application component to access an enterprise bean's home interface using EJB references.

To locate an enterprise bean's home interface, declare an EJB reference in the deployment descriptor and use JNDI to look up the interface. The referenced enterprise bean must be in the *ejb* subcontext of the application component's environment.

Declaring an EJB reference

You can declare an EJB reference in the deployment descriptor using the ejb-ref element. The data between the opening and closing ejb-ref tags defines an ejb-ref element. This code sample defines an EJB reference to the Employee entity bean:

```
<ejb-ref>
   <description>
      Reference to the Employee entity bean
   </description>
   <ejb-ref-name>ejb/Employee</ejb-ref-name>
   <ejb-ref-type>Entity</ejb-ref-type>
   <home>com.wooster.empl.EmployeeHome</home>
   <remote>com.wooster.empl.Employee</remote>
</ejb-ref>
```

An ejb-ref element contains:

- **description** This is optional.
- **ejb-ref-name** The name of the bean used in the application component.
- **ejb-ref-type** The bean type, Entity or Session.

- **home** The expected Java type of the home interface.
- **remote** The expected Java type of the remote interface.
- **ejb-link** This is optional.

This code sample illustrates how to use JNDI to look up the home interface of the Employee enterprise bean:

```
// Get the default initial JNDI context
Context initContext = new InitialContext();

// Look up the home interface of the Employee enterprise
// bean
Object result =
   initContext.lookup("java:comp/env/ejb/Employee");

// Convert the result to the correct type
EmployeeHome empHome = (EmployeeHome)
   javax.rmi.PortableRemoteObject.narrow(result,
                                    EmployeeHome.class);
```

Declaring an EJB local reference

To access an EJB's local interface, instead of the remote interface, define an EJB local reference (ejb-local-ref). Local interfaces are available only to EJB components, Java servlets, and JSPs hosted on the same server as the target component. This sample declares a local reference to the Employee bean, which provides access to its local interface:

```
<ejb-local-ref>
   <ejb-ref-name>ejb/EmployeeLocal</ejb-ref-name>
   <ejb-ref-type>Entity</ejb-ref-type>
   <local-home>
      com.wooster.empl.EmployeeLocalHome
   </local-home>
   <local>com.wooster.empl.EmployeeLocal</local>
   <ejb-link>Employee</ejb-link>
</ejb-local-ref>
```

Declaring an EJB link

You can define a link from an EJB reference to an enterprise bean by declaring an ejb-link element in the deployment descriptor. The application component and the target enterprise bean must be in the same J2EE application. This example creates a link to the Employee enterprise bean, by adding an ejb-link element to the bean's EJB reference definition:

```
<ejb-ref>
   <ejb-ref-name>ejb/Employee</ejb-ref-name>
   <ejb-ref-type>Entity</ejb-ref-type>
   <home>com.wooster.empl.EmployeeHome</home>
```

```
    <remote>com.wooster.empl.Employee</remote>
    <ejb-link>Employee</ejb-link>
</ejb-ref>
```

For information about using EAServer Manager to add and configure EJB references in Web applications, EJB components, and application clients, see "EJB references" in Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide*.

Resource factory references

A resource factory is an object that you use to create resources. You can assign a logical name to a resource factory in the deployment descriptor.

A resource-ref element defines a single resource factory reference. This code sample defines a reference to the resource factory that implements the DataSource interface:

```
<resource-ref>
   <description>
      Data source for the database in which the Employee
      enterprise bean records transactions
   </description>
   <res-ref-name>jdbc/EmployeeAppDB</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
</resource-ref>
```

A resource-ref element contains:

- **description** This is optional.
- **res-ref-name** Resource reference name used in the application's code.
- **res-type** Resource Java datatype that the application expects.
- **res-auth** Resource sign-on authorization, bean or container.

This code sample obtains a reference to the resource factory that implements the DataSource interface, and uses that reference to get a database connection (resource):

```
// Obtain the initial JNDI context
Context initContext = new InitialContext();

// Look up the resource factory using JNDI
javax.sql.DataSource ds = (javax.sql.DataSource)
   initContext.lookup
     ("java:comp/env/jdbc/EmployeeAppDB");
```

```
// Get a database connection
java.sql.Connection connection = ds.getConnection();
```

For information about using EAServer Manager to add and configure resource references in Web applications, EJB components, or application clients, see "Resource references" in Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide*.

UserTransaction references

J2EE application components can use the Java Transaction API (JTA) UserTransaction interface to manage transactions. A component instance can look up an object that implements the interface using the JNDI name *java:comp/UserTransaction.*

In this code sample, an application component uses the interface to manage a transaction:

```
// Get the initial JNDI context
Context initContext = new InitialContext();

// Look up the UserTransaction object
UserTransaction tran = (UserTransaction)
   initContext.lookup("java:comp/UserTransaction");

// Start a transaction
tran.begin();

// data updates

// Commit the transaction
tran.commit();
```

Configuring the EAServer naming service

Use the Naming Service tab on the Server Properties window to set a server's naming service options. You can use the Naming Service properties to configure a server to be a name server, or point to another server as its name server.

The Naming Service property sheet includes:

- This server's initial context.
- Whether or not this server is enabled as a name server.
- If this server is not a name server, the URL for the server that is acting as the name server.
- Heartbeat detection – periodically verifies that clustered name servers are either accepting client connections or have failed. See "Heartbeat detection" on page 129 for more information.
- If you are using an LDAP server to provide persistent name storage, the URL of the LDAP name server, as well as the manager DN (distinguished name) for the LDAP server and the manager's password.

For complete information about setting Naming Service properties, see "Naming Service" on page 33.

Name binding password security

You can establish password protection on the naming service to allow name binding only from a designated EAServer installation. This prevents unauthorized applications from creating name bindings using an EAServer name server.

To use the name binding password feature, you must set the com.sybase.jaguar.server.CosNaming.bindpassword property for the name server and each server participating in the naming service. You set this property using the Advanced tab in the Server Properties window. The default value is "jaguar."

All servers participating in the password-protected name service must have the same password as the name server. If the bindpassword property is empty, or does not exist in the property file for a name server, the name server accepts binds from any source.

Using an LDAP server with EAServer

To add persistent object name storage capabilities to EAServer, you can use an external directory naming service, such as an LDAP server. The server properties include an optional URL for specifying the port for the external name server.

When you use an external name server, EAServer uses JNDI to communicate with the name server through the specified URL.

LDAP object schema and EAServer objects

LDAP servers have predefined schema for common objects such as country, organization, and organizational unit. EAServer uses the following format for an LDAP-compatible initial context:

ou=<organizational unit>, o=<organization>, c=<country>

Storing EAServer object bindings on an LDAP server

When you use an LDAP server with EAServer, the *CosNaming* component binds all implemented objects on the servers that use the designated EAServer name server, and stores the name context information on the LDAP server. If EAServer detects previously-bound objects on the external name server, it updates the existing bindings with current name context information. When you shut down EAServer, it unbinds the stored information.

❖ **How the EAServer name server connects to an LDAP server**

1. On start-up, the EAServer name server connects to the LDAP server using the URL specified in the EAServer name server's Naming Service properties.
2. The EAServer name server authenticates the connection to the LDAP server using the manager DN specified in the EAServer name server's Naming Service properties.
3. The EAServer name server attempts to retrieve any existing matching name contexts from the LDAP server. If successful, EAServer uses the existing name context information.
4. The EAServer name server prepares the server object with the required attributes.
5. The EAServer name server attempts to add the server object to the LDAP server. If the object already exists, the LDAP server updates the existing object with the current attributes.
6. EAServer adds any new package/component name context information, or modifies the existing information if necessary.

CHAPTER 6

Clusters and Synchronization

A *cluster* is a group of servers that share replicated repository information. An EAServer cluster's primary purpose is to provide load balancing and high availability. See Chapter 7, "Load Balancing, Failover, and Component Availability" for more information. *Synchronization* enables you to connect to the primary server in a cluster and distribute repository information to "synchronize" one or more of the other servers in the cluster. You can also synchronize nonclustered servers. Synchronization provides a quick and easy way to distribute package, servlet, and other configuration information between servers.

Cluster overview

Each cluster includes a primary server, a group of participating servers, and a set of name servers:

- The *primary server* contains the master copy of the configuration repository for all servers in the cluster. The primary server distributes (synchronizes) its configuration to the other servers in the cluster.
- *Participating servers* or nonprimary servers share a "logical server name," which corresponds to a server defined in the primary server's repository. Several physical servers in a cluster share a logical server name; each like-named server shares components and servlets, and uses the same connection caches and other configuration information.

When you are configuring a cluster, you can use multiple logical server names to partition components. To ensure high availability, each logical server name must be shared by at least two physical servers in the cluster.

Note All servers within a cluster can share the same name as long as you are not partitioning.

- A cluster consists of at least two *name servers*. Each server in a cluster is aware of all of the name servers. Each server in a cluster binds its components to all name servers in the cluster. Binding all of the components of the clustered servers to multiple name servers provides high availability of your business components and redundancy if a server should go down, even if it is a name server.

  The list of name servers for the cluster override the naming properties for participating logical server definitions.

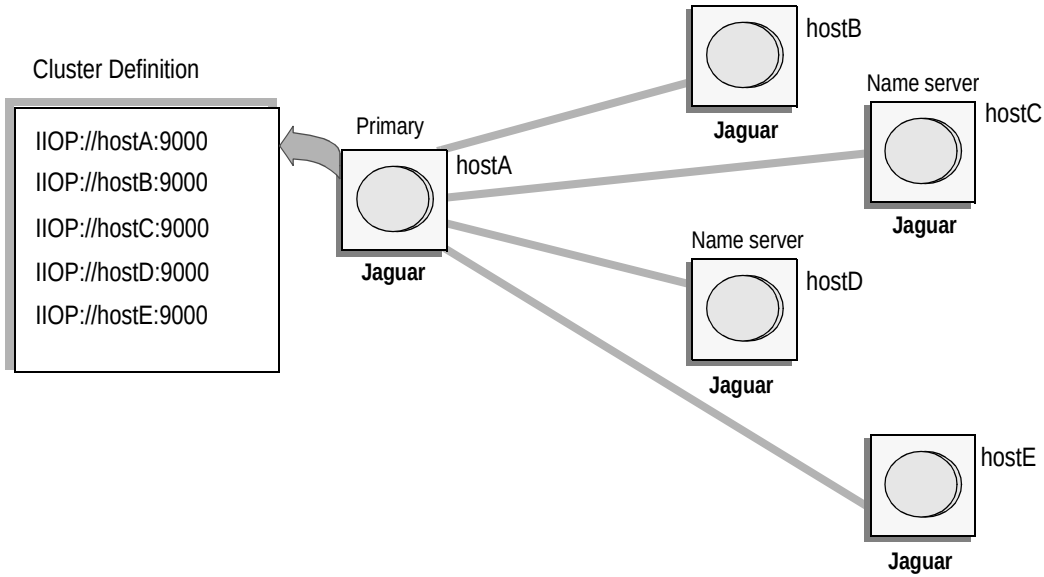
Cluster support is tightly integrated with the EAServer naming service, so that all client services from a cluster are made available through the naming service. See Chapter 5, “Naming Services” for more information.

Typically, each server in a cluster runs on a different host, so each server has its own copy of the entire repository and all files required for component execution. Sybase recommends that you run each cluster member from its own installation directory.

Cluster members can be on different platforms Beginning with EAServer version 4.0, a cluster can include servers on different platforms, including Windows, Solaris, AIX, HP, and Linux. Each server in the cluster must be the same version of EAServer. If the cluster uses PowerBuilder components, all the servers must use the same version of the PBVM.

Figure 6-1 illustrates a server cluster and uses hostA as the primary server to synchronize the participating servers, including the name servers. Each server in the cluster is named “Jaguar.”

Figure 6-1: EAServer cluster



Note The SessionManager::Session::lookup operation implicitly consults the naming service, so you can write a client that does not explicitly use the naming service but still takes advantage of cluster services. For example, the PowerBuilder connection object uses SessionManager::Session::lookup.

Cluster servers

Each server can be a member of only one cluster. To provide high availability, there should be at least two name servers defined for a cluster.

Servers are defined by URL, rather than by server name. Every server in a cluster can have the same name. If you are not using partitioning, it is easy to add a new machine to a cluster; simply change the host in each listener to the Internet host name or IP address. Then, connect EAServer Manager to the cluster's primary server and synchronize the cluster.

Do not use localhost listeners in clusters Servers running in a cluster cannot have localhost IIOP listeners. The sole exception is a cluster where all servers in the cluster run on one machine. In this case, all IIOP listeners must use the same host name, which can be localhost or the machine name.

A name server can be used by one or more clusters. See “Adding a name server to a cluster” on page 126 for restrictions before you assign a name server to more than one cluster.

All servers in a cluster should share a common account that will be used for inter-server connections when synchronizing the cluster. This must be a jagadmin account or an account for a user that has the Admin role.

If the servers in a cluster do not share a common account, you can synchronize the cluster only on a server-by-server basis.

Configuring a cluster

This section describes the steps required to create, configure, and manage a cluster using EAServer Manager.

❖ **Creating a cluster from EAServer Manager**

1 Highlight the Clusters folder.

2 Select File | New Cluster Wizard.

3 Enter the requested data on the Wizard pages; click Help for additional information on any page.

To set the initial context used by servers in a cluster when they bind their objects into the name servers, set the com.sybase.jaguar.cluster.initialcontext property on the Advanced tab in the Cluster Properties dialog. For example, cluster “US_Cluster” may have initial context “/US”, and cluster “UK_Cluster” may have initial context “/UK”. Then clients can talk to any name server which is used by either “US_Cluster” or “UK_Cluster”, and by specifying the appropriate initial context, (either “/US” or “/UK”) the clients can be directed to the appropriate servers.

❖ **Configuring a server to enable synchronization**

Before you can add a server to a cluster, you must configure each physical server to enable synchronization from the primary server.

1 Use EAServer Manager to connect to the physical server. If necessary, define a server that matches a logical server name defined in the primary server's repository. In other words, if the primary server's name is Jaguar_cluster, you must also assign the name Jaguar_cluster to each participating server.

2 If necessary, configure the listeners of this server. You must at least configure an IIOP listener to match the URL that will be used for cluster synchronization. Each like-named server in a cluster must also share the same listener name. Do not use 'localhost' for any IIOP listeners if the cluster runs on multiple machines.

3 Configure the account that will be used to synchronize the cluster:

- If you are using jagadmin, change the jagadmin password to match that of the primary server.
- If you are using an account other than jagadmin, add that account's user name or digital ID to the Admin role.

4 Restart the server so the new network addresses and security changes take effect.

❖ **Adding a server to a cluster**

Connect to the primary server with EAServer Manager and add each physical server to the cluster definition:

1 Click the Clusters icon.

2 Select the cluster to which you want to add servers.

3 Select File | Properties. The primary server (the server to which you are connected) displays in the Primary Server field of the General tab.

4 Select the Servers tab.

5 Click Add.

6 Enter a valid IIOP address for the server, for example iiop://myhost:9000. This address is used for inter-server connections when you synchronize the cluster.

When creating a server that will join the cluster, you must define and install listeners with the same names as used by the primary (in the master configuration) for that server name.

7 Once you have added all the servers that you intend to add to the cluster, synchronize the cluster. The Synchronize dialog box appears automatically after you add a server and click OK. See “Synchronizing a cluster from EAServer Manager” on page 135 for more information. You can add more servers later, but you must then again synchronize the cluster.

8 After you have synchronized the cluster, restart the new member servers. This step ensures that the new servers use the same security key information for interserver configuration.

Warning! After you add a nonprimary server to a cluster, EAServer Manager warns you when you connect directly to that server. Direct user updates to the server’s configuration can be overwritten when the cluster is synchronized if the new server has been the target of at least one synchronization before it was added as a member of the cluster.

❖ **Adding a name server to a cluster**

1 Click the Clusters icon.

2 Select the cluster to which you want to add name servers.

3 Select File | Properties.

4 Select the Name Servers tab.

5 Click Add.

6 Enter the server’s URL (for example, `iiop://myhost:9000`). When you have entered all of the name servers for the cluster, click OK. The Synchronize dialog box appears automatically after you add a name server and click OK. You can add more name servers later but must synchronize the cluster afterwards.

7 Synchronize the cluster.

Note Name servers can also be “ordinary” members of a cluster (if they are also listed on the Servers tab in the Cluster Properties dialog). However, if a name server is used by more than one cluster, it can be an ordinary member of only one cluster.

EAServer requires the cluster's bind password to authorize name context updates to the cluster's name servers. When you create a cluster, a random bind password is automatically generated. In most cases, you do not need to edit the password. However, if a name server is used by two or more clusters, you must configure the clusters to use the same bind password.

To change the password, modify the com.sybase.jaguar.cluster.bindpassword property on the Advanced tab in the Cluster Properties dialog—see com.sybase.jaguar.cluster.bindpassword on page 354. Sybase recommends that you use one of the randomly generated passwords, as security can be compromised if clients obtain knowledge of a cluster's bind password.

❖ **Removing a server from a cluster**

1 Click the Clusters icon.

2 Select the cluster from which to remove the server.

3 Select File | Properties.

4 Select the Servers tab.

5 Highlight the server you want to delete and click Delete.

6 Connect to the deleted server using EAServer Manager, and delete the com.sybase.jaguar.server.cluster property from the Server Properties dialog (this tells the server it is no longer a member of the cluster):

   a Double-click the Servers folder.

   b Highlight the server that has been deleted from the cluster.

   c Select File | Properties.

   d Select the Advanced tab.

   e Highlight the com.sybase.jaguar.server.cluster property.

   f Click Delete.

7 Synchronize the cluster, and restart the name servers.

❖ **Removing a name server from a cluster**

1 Click the Clusters icon.

2 Select the cluster from which to remove the name server.

3 Select File | Properties.

4 Select the Name Servers tab.

5 Highlight the server you want to delete and click Delete.

6 Synchronize the cluster.

❖ **Deleting an existing cluster**

1 Click the Clusters icon.

2 Select the cluster you want to delete.

3 Select File | Delete Cluster.

4 For each server that was in the cluster, verify that the server is no longer configured to join the deleted cluster, as follows:

   a Connect to the server with Jaguar Manager (if not connected already).

   b Highlight the server icon.

   c Select File | Properties.

   d Select the Advanced tab.

   e Delete the com.sybase.jaguar.server.cluster property.

A server that was part of the deleted cluster may try to connect to the cluster if the com.sybase.jaguar.server.cluster property is set to the name of the cluster. In these cases, you see error messages in the *server_name.log* file (where *server_name* is the name of the server) indicating that the server is in Admin mode. Fix this problem by deleting the property as described above.

❖ **Rebinding a cluster**

The Rebind option refreshes all of the name servers within a cluster. If you add a component to a server that is already part of a cluster and want to make that component available to the cluster, you need to rebind the cluster. You can also use the rebind option if a problem occurs when you synchronize the cluster; for example, if one of the name servers is slow to start.

1 Highlight the name of the cluster.

2 Select File | Rebind.

Heartbeat detection

The name servers in a cluster use heartbeat detection to periodically verify that member servers are either accepting client connections or have failed. If a server is not accepting connections, the name server does not return profile (host:port) information to the client, and routes requests to other servers in the cluster. The name server also detects when a failed server is ready to accept connections again and starts routing client requests to that server.

If a name server using transient storage fails, the cluster rebinds automatically when you reboot the failed name server. Otherwise, the cluster provides access to components through the remaining name servers in the cluster.

If a name server using persistent storage and LDAP support fails, the cluster does not need to rebind, but LDAP may leave behind stale profiles, resulting in unnecessary client retries and failures. For this reason, Sybase recommends you use transient storage to support load balancing and high availability.

❖ **Enabling heartbeat detection from EAServer Manager**

1 Select the Servers folder.

2 Highlight the name server for which you are configuring heartbeat detection.

3 Select File | Properties.

4 Open the Naming Service tab.

5 Select the Enable Heartbeat check box.

6 Enter the heartbeat frequency. This number is how often, in seconds, that the name server checks server availability. As the frequency period is shortened, server performance decreases. The default frequency is 120 seconds.

When you synchronize a cluster, the heartbeat settings (whether or not it is enabled and frequency) of the primary server are distributed to the other name servers in the cluster.

The com.sybase.jaguar.cluster.primary property stores the primary server URL for each cluster. The synchronization process maintains this automatically—see com.sybase.jaguar.cluster.primary on page 359. The cluster property, com.sybase.jaguar.cluster.version, stores the version number for each cluster. The synchronization process maintains this automatically—see com.sybase.jaguar.cluster.version on page 361.

Cluster start-up options

On start-up, a server that is a member of a cluster must verify that it is "in sync" with other cluster members by checking its cluster version number against other servers in the cluster. A server that is not in-sync remains in admin mode and does not join the cluster. The com.sybase.jaguar.cluster.startup cluster property specifies the testing algorithm. Set or modify this property from the Advanced tab in the Cluster Properties window—see com.sybase.jaguar.cluster.startup on page 360.

The com.sybase.jaguar.cluster.startup values and corresponding algorithms are:

- *check_primary* (default value) – With this setting, each server tests its cluster version number against the primary before joining the cluster, using the following algorithm:

  a If any other cluster member (including the primary) is reachable and has a higher cluster version number, this server moves to Admin mode since it is not "in sync."

  b If the primary is available and has the same cluster version number, this server joins the cluster and is ready to accept client connections.

  c If the primary is available and has a different cluster version number, this server moves to Admin mode since it is not "in sync."

  d If none of the above conditions are met, the server waits briefly, then returns to the first step in this algorithm.

- *check_servers* – With this setting, each server tests its cluster version number against the other member servers before joining the cluster, using this algorithm:

  a If any other cluster member (including the primary) is reachable and has a higher cluster version number, this server moves to Admin mode since it is not "in sync."

  b Let *M* be the number of cluster members (including the primary), and let *N* equal *M*/2 (integer division). If at least *N* other servers are available and have the same cluster version number, this server joins the cluster and is ready to accept client connections.

  c If none of the above conditions are met, the server waits briefly then returns to the first step in this algorithm.

  The *check_servers* option allows a server to join a cluster even if the primary is not available. This prevents a single point of failure if the primary server becomes unavailable.

- *disable_check* – no checking is performed. For a cluster with only two servers (including the primary), use this option and manually verify that the secondary server is "in sync." For a cluster with three or more servers, Sybase discourages the use of this option, since it can result in "out-of-sync" cluster members running together.

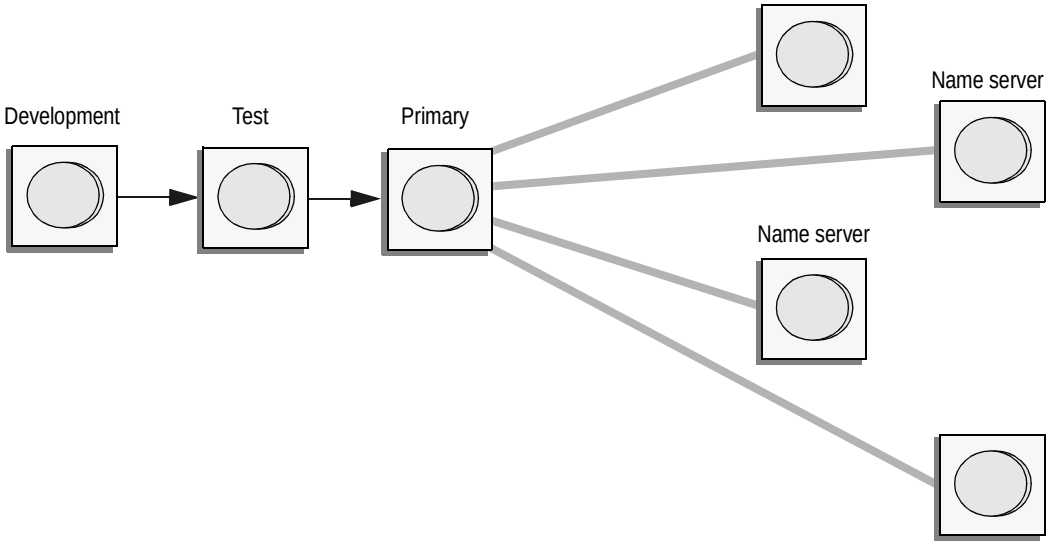
A server in Admin mode can be put into Ready mode as described in "Switching to Ready mode" on page 71. If a server is in Admin mode due to a cluster version number mismatch, then using Set Ready is the manual equivalent of *disable_check*, so using Set Ready in this case is also discouraged as it may result in "out-of-sync" cluster members running together. Sybase recommends that you synchronize the cluster instead. See "Synchronizing a cluster from EAServer Manager" on page 135 for more information.

Synchronization

Synchronization replicates application files and configuration information between servers. If you are using clusters, synchronization ensures that logical servers in a cluster share the same application files and configuration. Synchronization is also a useful alternative to importing and exporting packages.

For example, as Figure 6-2 illustrates, you can replicate new components from a testing or development server to one or more production servers. As long as you can connect to both servers, synchronization is quicker and easier than exporting and importing package files.

Figure 6-2: Synchronization example



If you are using clusters and make configuration changes to the primary server, the jagadmin user (or any user with the Admin role) can synchronize those changes to participating nonprimary servers.

To synchronize a cluster, you must be connected to the primary server for the cluster unless the primary server is down and cannot be restarted. In this case, you can connect to another server within the cluster and designate it as the new primary server.

You can use the synchronization option to move repository information between servers that are not part of a cluster. For example, you may want to move a development server's repository to a test server. In this case, the synchronization command is similar to EAServer's export/import options without the JAR file; instead of moving packages from one server to another, you are moving repository information.

Synchronization does not propagate deletions. If you delete an entity such as a component or Web application in the source installation, and you have previously replicated the entity by synchronization, you must connect to the target server with EAServer Manager or jagtool and delete the entity manually.

You can synchronize at the cluster, server, servlet, package, or component level. The level you choose determines the available synchronization options. Options that are not available are dimmed, and cannot be selected.

Note Before you synchronize a cluster, shut down any services that are running. If you do not, some files required for synchronization may be locked.

Component synchronization

The items that are synchronized for a component are the same in a Java archive (JAR) file, as for a package when using EAServer's Export feature, and include:

- Component definition (component properties)
- Interface definition(s)
- For C/C++ components, the DLL (or shared library) indicated by the com.sybase.jaguar.component.cpp.library property
- For Java and EJB components, the implementation class (property com.sybase.jaguar.component.java.class), any classes referred to by the com.sybase.jaguar.component.java.classes property, plus stub classes listed in the com.sybase.jaguar.component.files.corbastubs, and com.sybase.jaguar.component.files.ejbstubs properties
- For PowerBuilder components, the libraries starting with a "$" that are referenced by the com.sybase.jaguar.component.pb.librarylist property

  Note If you are synchronizing PowerBuilder components, the servers must all use the same PBVM version.

- Other files indicated by the com.sybase.jaguar.component.files property.

Synchronizing after deleting or moving components Synchronization does not remove components that you have deleted. You must connect to the target server and delete the component with EAServer Manager or jagtool. If you move a component from one package to another, you must delete it from the original package on the target server.

Package synchronization

When you synchronize an EAServer package, these items are distributed to the remote servers:

- The same items as for component synchronization for each component in the package.
- Java stub classes indicated by com.sybase.jaguar.package.files.corbastubs and com.sybase.jaguar.package.files.ejbstubs properties.
- Other files indicated by property com.sybase.jaguar.package.files.

Servlet synchronization

When you synchronize an EAServer servlet, the following items are distributed to the remote servers:

- The servlet definition (servlet properties).
- The servlet class (property com.sybase.jaguar.servlet.java.class), and any classes referred to by property com.sybase.jaguar.servlet.java.classes.
- Other files indicated by property com.sybase.jaguar.servlet.files.

Application synchronization

When you synchronize an EAServer application, the following items are distributed to the remote servers:

- The application definition (application properties)
- The packages, Web applications, and the other resources contained in the application

Web application synchronization

When you synchronize an EAServer Web application, the following items are distributed to the remote servers:

- The Web application definition (Web application properties).
- The servlets, JSPs, and filters installed in the Web application, including implementation classes and properties.
- By default, the Java classes listed in the Web application's custom class list specified on the Java Classes tab in the EAServer Manager Web Application Properties dialog box. You can disable synchronization of these files in the synchronization options described in Table 6-1 on page 136.
- The Web application's context root directory and its contents.
- Files specified on the Additional Files tab in the Web Application Properties dialog box in EAServer Manager (or by the com.sybase.jaguar.webapplication.files property if using jagtool).

Synchronization procedures

You can synchronize servers and application components from EAServer Manager or by using the jagtool sync command.

❖ **Synchronizing a cluster from EAServer Manager**

1 Use EAServer Manager to connect to the source (primary) server.

2 Depending on the level of the sync operation, highlight a cluster, server, package, servlet, or component.

> **Note** Synchronization is enabled for servers, packages, and components for any EAServer edition. Cluster synchronization is enabled only for the Enterprise Edition. The cluster options display in the Synchronize dialog for any object, but are disabled if the Enterprise Edition is not licensed.

3 Select File | Synchronize.

4 Supply the synchronization information in the Synchronization dialog and click Start Sync. Table 6-1 on page 136 describes the synchronization properties.

A server is in Admin mode if it is running and accepting connections from EAServer Manager, but not accepting ordinary client connections. A server in Admin mode can be the target of synchronization, since this is often the appropriate way to get it out of Admin mode. EAServer Manager warns you if you connect to a server that is running in Admin mode. You can force a server into Ready mode as described in “Switching to Ready mode” on page 71, but you should understand why the server is in Admin mode before you use this option.

Table 6-1: Synchronization properties

Property	Description	Comments
Username	The user name used to log in to the remote servers.	The user must belong to the Admin Role to use the synchronize option.
Password	The password for the user name on the remote servers.	When synchronizing to multiple remote servers, including cluster members, all remote servers must use the same password to allow connection to those servers.
Cluster	Click the Cluster check box and select a cluster from the drop-down list to synchronize the entire cluster.	If you are synchronizing a server, package, component, or servlets, you can specify that the changes be replicated to all servers in a cluster. Use this option only when connected to the cluster's primary server, or when connected to a nonprimary server that becomes the primary after synchronization.
Servers	Click the Servers check box and supply a list of one or more URLs to be targets of synchronization if you want to synchronize a subset of the cluster.	The URLs must be separated by a comma, and of the form `iiop://hostname:port`.
All Web App Files	When this option is selected, all Web application files are distributed to the remote servers.	See “Web application synchronization” on page 135 for more information.
Synchronize Java Classes	Applies to Web application synchronization only. Specifies whether to synchronize Java classes listed in the Web application’s custom class list specified on the Java Classes tab in the EAServer Manager Web Application Properties dialog box.	
All Package Files	When this option is selected, all of the package files are distributed to the remote servers. Package files refers to all user-defined IDL files, property files, and implementation files (Java .class, .dll, .so, .pbd) based on component definitions.	Selecting this option may result in the synchronization taking a long time, depending on the number of files being transferred to the remote server(s). Consider using component-level synchronization if only a few components have been changed. See “Component synchronization” on page 133 and “Package synchronization” on page 134 for more information.

Property	Description	Comments
All Servlet Files	When this option is selected, all servlet files are distributed to the remote servers.	See "Servlet synchronization" on page 134 for more information.
All App Files	When this option is selected, all application files are distributed to the remote servers.	See "Application synchronization" on page 134 for more information.
All Connector Files	When this option is selected, all connector files are distributed to the remote servers.	
All Files Matching the Wildcard Pattern	When this option is selected, all files that match the wildcard pattern you enter in the adjacent field are distributed to the remote servers.	
Verbose	Run in verbose mode.	
Refresh	If you select this option, the servers you are synchronizing refresh when synchronization completes.	
Restart	If you select this option, the servers you are synchronizing restart when synchronization completes.	A timeout is used so that if a remote server does not restart, the synchronization moves to the next server.
Then Wait	This is the amount of time that the primary waits for a server to restart if you have selected the Restart option.	

❖ **Overriding synchronization operations**

You may want to manually configure properties on individual servers in a cluster to override properties from the logical server definition. For example, you may want to:

1 Increase the size of a connection cache on a machine that has more memory than other machines.

2 Establish a process for making manual updates after each EAServer Manager synchronization.

CHAPTER 7

Load Balancing, Failover, and Component Availability

This chapter discusses:

- Load balancing – optimizes performance for your EAServer cluster by adjusting the load across the servers.
- Component deployment – you can restrict access to components by deploying them to a subset of servers within a cluster, or make them available from all servers.
- High availability – an EAServer cluster provides redundancy (high availability) of business components and services in case a server within a cluster fails.
- Automatic component failover – allows a client's object reference to be usable across servers should a server within a cluster fail.
- Sybase Failover for high availability systems – you can implement the Adaptive Server Enterprise failover feature with EAServer database connectivity using Java Connection Management (JCM).

To enable these features, you must first create a server cluster (a group of servers running on different machines). The servers in a cluster share the workload and provide client services even if one or more servers within the cluster fails or is offline. See Chapter 6, "Clusters and Synchronization" for information about creating a cluster.

Understanding load balancing

Load balancing in an EAServer cluster is determined by three factors, each of which is discussed in detail in this section:

- Load metrics – if you select a dynamic load policy, the load metrics determine the load on your servers and give each server a numerical weighting, which is then used to distribute incoming client requests and optimize performance of the cluster.
- Load distribution policy – when you configure load balancing, you select the distribution policy that best matches your environment and situation.
- Interoperable object reference (IOR) – contains a profile that the client uses to look up a component. The profile contains the server and port number that the client uses to access the component. The distribution policy determines the order in which the profiles are distributed to the clients.

Load metrics

Load metrics is a collection of system statistics that define a load on a server. Each server in a cluster is assigned a value, or **normalized load**, based on the load metrics.

Many factors affect overall system performance and throughput of a server. The load metrics that EAServer uses to determine the normalized load are:

- CPU utilization
- Method response time
- IIOP connections

When overall cluster load is light, incoming requests are evenly distributed to all member servers; that is, all member servers have some load. When the cluster becomes more loaded, the load is distributed according to a server's current load.

How load metrics work

Various system load metrics are collected at each **sampling interval** by the **load collector** in each server. At each **broadcasting interval**, the load collector broadcasts its load metrics to all member servers. At each **calculating interval** the load collectors in the name servers calculate and generate a **normalized load list** (NLL) of all member servers sorted according to their load metrics. The NLL consists of two elements: the server name and normalized load number. For example, a three-server cluster may have an NLL of:

(Jaguar_server_A, 5), (Jaguar_server_B, 3), (Jaguar_server_C, 1)

Which indicates that the load of Jaguar_server_A is five times greater than the least-loaded server, Jaguar_server_C.

A normalized load number ranges from 1 to 5 and is generated by combining all weighted load metrics. A value of 5 indicates a heavily loaded server, while a 1 indicates a lightly loaded server.

All name servers have their own copies of the NLL, which reflects the Naming Service's view of system-wide load distribution that remains static for an entire refresh interval.

The NLL is referenced to balance the load of a cluster if an adaptive load distribution policy is used. See "Load distribution policies" on page 141 for information about the various distribution policies.

Load distribution policies

There are four load distribution policies:

- Random – static, even distribution.
- Round-robin – static, even distribution.
- Weighted – static, random based on server weight.
- Adaptive – dynamic, random based on load metrics.

The first three policies do not rely on load metrics, so there is no need to obtain an NLL.

The naming service applies the selected load distribution policy and generates an IOR with multiple profiles. See "Interoperable object references" on page 142 for more information.

Random and round-robin policies

Both random and round-robin policies attempt to evenly distribute incoming client requests to all participating servers. These distribution policies are ideal for a cluster with comparable physical nodes that have similar performance characteristics.

Weighted policy

The weighted policy allows you to specify the processing load to each server. This policy is designed for physical nodes with a wide range of performance variations. You can specify any weight between 1and 5 for each server. A larger number indicates that the server can handle a larger load.

Adaptive policy

The adaptive policy is similar to the weighted policy but adapts for runtime load variation. It examines the NLL for current load metrics, and compensates for uneven loads by distributing proportionately more client requests to lightly loaded servers. There is more overhead with the adaptive policy than the other policies.

Load collectors reside in each server, and the failure of any load collector has no global effect. If a broadcast of load metrics from a particular load collector has not been received for a predefined period of time, the corresponding server is dropped from the NLL.

The load distribution is fully synchronized and is linked to the name servers as part of the naming services. The high availability protection for naming services also applies to dynamic load balancing. See “Understanding high availability” on page 143 for more information.

Interoperable object references

Load balancing uses EAServer’s naming service to distribute incoming IIOP requests across the servers within an EAServer cluster:

1 The client obtains a factory IOR from the name server when it performs a lookup operation on a component. This factory IOR contains a profile (a server::port combination) that identifies the servers from which the component is available. There is a profile for each server, or if a single server has multiple IIOP ports, a profile for each port.

   The name server uses the selected load distribution policy and generates an IOR with multiple profiles to balance the requests between available servers and ports. If a dynamic load policy is selected, the NLL is used to determine and balance the load of the individual servers.

2 Using the factory IOR, the client contacts a server using one of the profiles to obtain the IOR for the component. The IOR for the component has only one profile within it.

3 The client sends the IIOP request to the profile specified by the IOR of the component.

See the *EAServer Programmer's Guide* for detailed information.

Understanding high availability

High availability provides access to your business components and services even if a server is unavailable.

You can use clusters to achieve high availability if:

- Each cluster has at least two member servers.
- Partitioning is used, and each partition has at least two member servers.
- Each cluster has at least two name servers (which either can be members of the cluster or can be external to the cluster).

To guarantee end-to-end high availability, clients should use URLs of the form "iiop://host1:9000;iiop://host2:9000;..." when doing any of the activities listed below.

In the first three cases, the client's URL list should be the cluster's server list, or a subset of that list.

- Setting the location property for the PowerBuilder connection object.
- Setting the URL for the creation of a SessionManager::Manager object (for C++ and Java CORBA clients).
- Setting the InitialContext Provider URL property for Java clients using JNDI.
- Setting the NameServiceURL during ORB initialization. In this case, the client's URL list should be the cluster's name server list, or a subset of the list.

This ensures that no part of a client's initialization is limited to a single point of failure.

Note You can support non-EAServer clients (that do not support the EAServer-proprietary multiple URL form) by creating an EAServer service component that, upon server start-up, writes a file containing the stringified IOR for a multiserver URL. This IOR file can then be read by any client using an HTTP connection.

Configuring load balancing

Configure load balancing in EAServer Manager. While the cluster is running, you can display the per-server load to test the load balancing configuration.

❖ **Configuring load balancing**

1 Select the Clusters folder.

2 Select the cluster that you want to configure.

3 Select File | Properties. The Cluster Properties window displays.

4 Select the Load Balancing tab. Click the Dynamic Load Balancing Enabled check box.

5 Complete the load balancing property sheet and click OK. Default values are provided but you may want to modify them depending on the capabilities of the servers in your cluster. Load balancing properties are described in Table 7-1 on page 145.

Table 7-1: Load balancing properties

Property	Description	Comments
Dynamic Load Balancing Enabled	Enables configuration of the dynamic load balancing policies (Weighted or Adaptive).	
Sample interval	The interval, in seconds, that each server collects its load metrics.	The default value is five seconds.
Broadcast interval	The interval, in minutes, that each server broadcasts its load metrics to the other servers in the cluster.	The default value is five minutes.
Calculate interval	The interval, in minutes, that the load collectors in the name servers calculate and generate a normalized load list (NLL) of all member servers.	The default value is ten minutes.
Refresh interval	The interval, in minutes, that each name server obtains the NLL from its local load collector. All name servers have their own copies of the NLL. The refresh interval is equal to or greater than the calculate interval.	The default value is ten minutes.
Maximum weight	The maximum weight, used in a weighted load balancing policy, of any server.	The default maximum weight is five. The maximum value is ten.
Policy	From the drop-down list, select the load balancing policy for this cluster.	If you select a weighted policy, you must supply a weighted value for each server in the cluster.
Weights	This property is enabled if you selected a weighted policy. Supply a weighted value for each server in the cluster, not to exceed the maximum weight value.	See “Weighted policy” on page 142 for more information.

❖ **Viewing the current per-server load**

1 Select the Clusters folder.

2 Highlight the cluster you want to view.

3 Select File | Display Cluster Load.

The load graph is a bar chart. The load of each server is expressed as a percentage. Load is defined as the ratio of the number of bytes sent in the last 1/2 second over 10,000, expressed as a percentage.

The status list contains an entry for each server and tells whether the server is in Ready or Admin mode. If a server is in Admin mode, the list tells why.

Deploying components to a cluster

To deploy components on a cluster so that every component is available from every cluster member use the synchronization feature–see “Synchronization” on page 131. The sections below describe additional configuration that is required for components deployed in a cluster.

Stateful components

Stateful components such as EJB stateful session beans require additional configuration when deployed in a cluster. You must configure persistent state storage or inter-server replication so that the instance state is available on the different servers in the cluster. For details, see Chapter 28, “Configuring Persistence for Stateful Session Components,” in the *EAServer Programmer’s Guide*.

Partitioned components

Partitioning restricts components to a subset of servers within a cluster. Partition an application by creating different logical server names, then installing component subsets into each logical server definition. Partitioning allows you to manually load balance within your cluster.

The Installed Packages folder lists the packages that are installed in a particular server. Even if a package is defined in a server's repository, and all files required for components in the package are available to the server, the server does not allow creation of instances of components for packages that are not installed in the server.

The following is a partitioning example:

- Cluster “MyCluster” has three members: `iiop://host1:9000;iiop://host2:9000;iiop://host3:9000`
- Server `iiop://host1:9000` has package “P1” installed.
- Server `iiop://host2:9000` has package “P2” installed.
- Server `iiop://host3:9000` has packages “P1” and “P2” installed.
- Package P1 is available from `iiop://host1:9000` and `iiop://host3:9000`, and package P2 is available from `iiop://host2:9000` and `iiop://host3:9000`.

See Chapter 3, “Managing Applications and Packages in EAServer Manager,” in the *EAServer Programmer’s Guide* for more information about installing packages.

Note You may want to use partitioning to separate CPU-bound components from database-bound components.

Automatic failover for components

You can use EAServer Manager to mark selected components to support transparent automatic failover. If a client has an object reference for a component on a server that is a member of a cluster, the client’s object reference will provide transparent failover, unless all the servers in the cluster fail.

Note To avoid a single point of failure for a cluster, set the com.sybase.jaguar.cluster.startup cluster property to “check_servers.” See “Cluster start-up options” on page 130 for more information.

Automatic failover is not the default for EAServer components. When a client is using a component that does not support automatic failover, and the server hosting that component fails, the client must create a new instance of that component to recover from the failure (which typically presents itself as a CORBA COMM_FAILURE system exception). However, the client does not need to create a new session, since the SessionManager::Session object supports automatic failover. The SessionManager::Session object is used implicitly by the PowerBuilder connection object and by the EAServer COM (ActiveX) proxy.

❖ **Setting automatic failover for a component from EAServer Manager**

1 Locate and highlight the component you want to set.

2 Select File | Properties.

3 Select the Transactions tab.

4 Select the Automatic demarcation/deactivation check box and the Autofailover check box.

5 Click OK.

Component implementation guidelines

The following guidelines may be useful when you are writing components that support automatic failover.

The component should not retain conversational state in server memory (component instance variables), since the conversational state cannot be restored when a remote method call fails over from one server to another.

The following example shows why this would not work:

1 The client calls method A on component C on Server1. Method A retains some state in the instance in Server1's memory.

2 The client calls method B on the same component. Server1 has failed, so the client transparently fails over to Server2 and calls method B on a newly instantiated instance of component B in Server2. Since method A has not been called on this instance, it does not hold the saved state.

If you must save state between calls, consider saving it in a database. For example, in an Internet shopping application, a "shopping cart" might be represented by a database entity, and every method call on the ShoppingCart component can save the appropriate changes to the database.

In other cases, you might want to code the client to use IDL structure and sequence types to pass a list of values to a single component method, instead of passing each value in a separate call and having the component attempt to collect the list of values using conversational state. This approach also reduces network traffic, and can greatly improve response times.

Duplicate database inserts or updates can result from the use of automatic failover, as in the following example:

1 The client calls method insertStuff on component C on Server1.

2 The insertStuff method inserts a record into a database.

3 The transaction is committed.

4 The server crashes before sending the reply message over the network to the client.

5 The client transparently fails over, and calls method insertStuff on a new instance of component C on Server2.

6 The insertStuff method inserts a new (duplicate) record into the database.

Everything works this time, but we now have a duplicate record in the database.

A simple design approach can help avoid such problems. Add a method to component C to generate a new ID for insertion: for example, newStuffId:

1 The client calls newStuffId to get a new unique ID. If you do not permit gaps in the ID numbering sequence, you cannot use this approach.

2 The client then calls insertStuff, passing the *StuffId* as a parameter.

3 insertStuff verifies that a record for that *StuffId* has already been inserted (or the database insert fails if *StuffId* is a unique key in the database).

Although insertStuff has been called twice, only one database change has been made.

A component that supports automatic failover can use other components or resources that do not directly support automatic failover.

Deploying Web applications to a cluster

Web applications can be distributed by deploying them to an EAServer cluster. A distributed Web application can provide better performance since multiple machines can handle more load than one. Clusters also provide high availability: if one machine goes off-line, clients can connect to another server in the cluster.

Clustered Web application requirements

To deploy your Web application in a cluster, you must have a mechanism to support load balancing (and optionally failover), configure a mechanism to replicate HTTP session data between servers in the cluster, and make sure your code supports distributed deployment.

Load balancing and failover

Since the HTTP protocol does not support failover and load-balancing, you must configure a system to redirect client requests that use one logical host name to the cluster servers. You can do this using one of the following solutions:

- Use the EAServer Web server redirector plug-in, running in Apache or another supported Web server. For information on this option, see the *EAServer Installation Guide* for your platform. This option allows load-balancing, but not high availability. The Web server can be a single point of failure in your configuration.
- Use Round Robin DNS (RRDNS). RRDNS is a standard feature in many operating systems, and no extra hardware is required. RRDNS allows HTTP requests to be routed in a round-robin fashion to different Web servers. For information on the advantages and disadvantages of RRDNS, see the O'Reilly article Web Applications Load Balancing at http://www.onjava.com/pub/a/onjava/2001/09/26/load.html. RRDNS provides load balancing, but not high availability. The RRDNS service can be a single point of failure.
- Use another third-party address-redirection system that performs HTTP load-balancing and failover, such as:
  - The BIG-IP hardware load redirector, from F5 Networks at http://www.f5.com/
  - The Local Director hardware load redirector, from Cisco Systems at http://www.cisco.com/
  - If you have a cluster of Windows 2000 Advanced servers, you can use built in features for network load balancing across a cluster. Similar functionality is available for NT. For more information, see the Microsoft MSDN article Building a Highly Available and Scalable Web Farm at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnduwon/html/d5nlb.asp.

Session data replication

If a Web application is distributed and running in a cluster, EAServer replicates session data to all servers in the cluster, using one of the following mechanisms:

- *Persistent storage*: EAServer stores session data in a persistent data store to support shared sessions and session failover.
- *In-memory replication*: EAServer replicates session data between pairs of servers, each of which acts as a backup for the other. This feature can improve performance by avoiding the overhead of writing to the database.

You must configure one of these options, as described below.

Coding considerations

No changes are required to your servlet and JSP implementation code to support distributed sessions, as long as:

- You are managing session data using the servlet session APIs or some other mechanism where storage is not tied to the host server (such as an EJB session or entity bean).
- You use a database (or an EJB entity bean that connects to a database) to store global data. You can use the Web application's environment properties to store global read-only data.

Since session data is bound to a single user, you cannot use sessions to store global read-write data. Many applications use ServletContext properties to store global data, but the ServletContext is not global to a distributed application and cannot be used as a read-write shared-memory store.

Configuring in-memory HTTP session replication

EAServer can distribute HTTP session data using in-memory replication rather than database storage. This feature can improve performance by avoiding the overhead of writing to the database. This mechanism uses the mirror-pair replication model described in "Requirements for in-memory stateful failover" in Chapter 29, "Configuring Persistence Mechanisms," in the *EAServer Programmer's Guide*.

❖ **Enabling in-memory replication for a Web application**

These steps must be performed in EAServer Manager, while connected to the primary server for your EAServer cluster:

1 Select the Distributed checkbox on the General tab in the Web Application Properties dialog box.

2 Install the Web application to one or more logical servers that are part of the cluster.

3 Configure mirror pairs for the cluster as described in "Cluster configuration for in-memory failover" in Chapter 29, "Configuring Persistence Mechanisms," in the *EAServer Programmer's Guide*.

4 On the All Properties tab, set the com.sybase.jaguar.webapplication.distribute.type property to "inmemory".

5 Synchronize the cluster.

❖ **Changing the cache size**

The default cache size and entry time out values are unlimited. To change these settings:

1 In the master server installation for your cluster, create the directory *ObjectCache* in the *Repository* directory your EAServer installation if it does not exist.

2 In this directory, create a text file named *HttpSessionCache.props* if it does not exist.

3 Edit *HttpSessionCache.props* in a text editor, and enter the following lines:

```
com.sybase.jaguar.objectcache.name=HttpSessionCache
com.sybase.jaguar.objectcache.size=size
com.sybase.jaguar.objectcache.timeout=timeout
com.sybase.jaguar.objectcache.sync=mirror
```

Where *timeout* is the timeout value, in seconds, and *size* is the size in megabytes, kilobytes, or bytes with the syntax shown in the following table:

Syntax	To indicate
*n*M or *n*m	*n* megabytes, for example: `512M`
*n*K or *n*k	*n* kilobytes, for example: `1024K`
n	*n* bytes, for example: `536870912`

4 Synchronize the cluster to apply the changes to other member servers.

Configuring persistent session storage

When using this option to replicate session data, EAServer stores all session data in a remote database, connecting through the predefined JDBC connection cache ServletPersistenceCache. All servers in the cluster share the same database. Sybase recommends that you configure this cache to connect to an enterprise-grade database server. The database cannot be shared between servers that are not running in the same EAServer cluster.

The sample ServletPersistenceCache properties must be changed As preconfigured, the ServletPersistenceCache connects to the sample database that is included with the EAServer sample applications. This sample uses the evaluation version of Adaptive Server Anywhere, which does not allow connections from multiple hosts. You must use another database that allows connections from multiple hosts, and supports the number of connections required by your cluster.

❖ **Configuring persistent session storage**

These steps must be performed in EAServer Manager, while connected to the primary server for your EAServer cluster:

1 Select the Distributed checkbox on the General tab in the Web Application Properties dialog box.

2 Install the Web application to one or more logical servers that are part of the cluster.

3 Configure the properties of the connection cache named ServletPersistenceCache to connect to the database that you use for persistent session storage. This cache must use JDBC and have cache-by-name access allowed. See Chapter 4, “Database Access,” for instructions.

4 Make sure the ServletPersistenceCache cache is installed in each logical server where the Web application is installed.

5 If using a database other than Sybase Adaptive Server Enterprise or Adaptive Server Anywhere, create a table as described in “Creating the database table” on page 154.

6 Synchronize the EAServer cluster to propagate the configuration changes to other servers in the cluster. See “Synchronization” on page 131 for more information.

Creating the database table

If you are storing session data in a database other than Sybase Adaptive Server Enterprise or Adaptive Server Anywhere, you must manually create the table that stores the session data. Create a table named ps_HttpSession with the following schema:

Column	Data format
ps_key (primary key).	Variable length binary, 255 bytes maximum length, cannot be null.
ps_size	Integer, cannot be null.
ps_bin1	Variable length binary, 255 bytes maximum length, can be null.
ps_bin2	Variable length binary, 255 bytes maximum length, can be null.
ps_bin3	Variable length binary, 255 bytes maximum length, can be null.
ps_bin4	Variable length binary, 255 bytes maximum length, can be null.
ps_data	Binary large object. This type must be functionally equivalent to a Sybase image type. The JDBC driver used by the specified connection cache must allow access to the ps_data column using the JDBC setBytes and getBytes methods.

The following table definitions can be used for creating an Oracle 8.1.7 database:

```
PS_KEY   RAW (255)     NOT NULL,
PS_SIZE  NUMBER        NOT NULL,
PS_BIN1  RAW (255),
PS_BIN2  RAW (255),
PS_BIN3  RAW (255),
PS_BIN4  RAW (255),
PS_DATA  LONG RAW
```

Implementing Sybase Failover for high availability systems

You can implement the Sybase Failover feature in Adaptive Server Enterprise 12.0 and later with EAServer database connectivity using either Java Connection Management (JCM) or the Sybase Open Client Client-Library.

Java Connection Management

You configure JCM by enabling Java Database Connectivity (JDBC) connections to establish failover-enabled connections to Adaptive Server Enterprise. jConnect requires that the connection's attributes be configured on an LDAP server and accessed using JNDI. See the jConnect and JNDI documentation for more details on the jConnect configuration for LDAP.

You may need to configure your LDAP server to accept the Sybase schema extensions. You can find the schema definitions in the *sybase.schema* file located in the *$SYBASE_OCS/config* directory of your Adaptive Server installation. For information on configuring your LDAP server, see your LDAP server documentation.

You can implement failover using:

- JDBC 1.0 and 2.0, or
- JDBC 2.0 extension/JTA drivers

JDBC 1.0 and 2.0

Set up jConnect to access LDAP and JNDI with JDBC 1.0 and JDBC 2.0 connections by:

- Configuring LDAP
- Configuring a connection cache

Configuring LDAP

To configure LDAP, run your server with JDK 1.3 and use your LDAP software to set up an LDAP entry of this form. In this example, *primary_server* is the name of the high availability (HA) primary server, *secondary_server* is its secondary server, and *host* and *port* are the host name and port number of the machine on which the database server runs.

```
dn: sybaseServername=primary_server, dc=sybase, dc=com
sybaseServername: primary_server
sybaseAddress: TCP#1#host port
sybaseHAservername: sybaseServername=secondary_server
sybaseJconnectProperty: REQUEST_HA_SESSION=true
sybaseJconnectProperty: Tds
objectclass: sybaseServer

dn: sybaseServername=secondary_server, dc=sybase, dc=com
sybaseServername: secondary_server
sybaseAddress: TCP#1#host port
objectclass: sybaseServer
```

Configuring a connection cache

Use EAServer Manager to configure the connection cache with these values for the specified properties:

- User Name – the database user name.
- Password – the database password.
- Connection URL (Server Name) – `jdbc:sybase:jndi:ldap:sybaseServername=`***`primary_server`***

To specify additional properties that EAServer uses for LDAP server configuration, create a *<cache_name>.props* file in the *$JAGUAR/Repository/ConnCache* directory, and set these property values:

```
java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
java.naming.factory.object=com.sybase.jdbc2.jdbc.SybObjectFactory
java.naming.provider.url=ldap://duplo:389/dc=sybase,dc=com
```

where *duplo:389* is the port where the LDAP server is running. The values listed for the first two properties are the defaults used by EAServer.

JDBC 2.0 extension/JTA drivers

Set up JDBC 2.0 extension / JTA drivers by:

- Configuring LDAP
- Configuring an XA resource

Note When failover occurs, EAServer supports XA transactions only if they are in the **prepared** state. XA transactions in any other state are lost. For information on transactions, see Chapter 2, "Understanding Transactions and Component Lifecycles," in the *EAServer Programmer's Guide*.

Configuring LDAP

Using your LDAP software, set up an LDAP entry of this form:

```
dn: sybaseServername=primary_xa_server,dc=sybase,dc=com
sybaseServername: primary_xa_server
sybaseAddress: TCP#1#host port
sybaseHAservername: sybaseServername=secondary_xa_server
sybaseJconnectProperty: REQUEST_HA_SESSION=true
sybaseJconnectProperty: Tds
objectclass: sybaseServer

dn: sybaseServername=secondary_xa_server,dc=sybase,dc=com
sybaseServername=secondary_xa_server
sybaseAddress: TCP#1#host port
objectclass: sybaseServer
```

Configuring an XA resource

Use EAServer Manager to configure the connection cache with these values for the specified properties:

- User Name – the database user name.
- Password – the database password.
- Connection URL (Server Name) – `LDAPLOOKUP=sybaseServername=`***`primary_xa_server`***.
- Driver (DLL or Class Name) – com.sybase.jdbc2.jdbc.SybXADataSource.

Create a *<cache_name>.props* file in the *$JAGUAR/Repository/ConnCache* directory, and set this property value, where *duplo:389* is the port where the LDAP server is running:

```
java.naming.provider.url=ldap://duplo:389/dc=sybase,dc=com
```

Troubleshooting the database connection

To find out if your database connection to the LDAP server is working, ping the connection cache.

❖ **Pinging the connection cache**

- Right-click the connection cache, and select Ping.

If the server does not respond or an error occurs, verify that:

- The CLASSPATH and BOOTCLASSPATH specify the correct locations for the *ldap.jar*, *providerutil.jar*, *jconn2.jar* (the jConnect driver), and *jndi.jar* (JNDI 1.2)

- You are using JDK 1.3.

Open Client Client-Library

You can establish high availability Client-Library connections to an Adaptive Server Enterprise database by:

- Modifying the client connection information
- Selecting the high availability option
- Using the connection APIs

Modifying the client connection information

On UNIX platforms, set these values in your *interfaces* file:

Server1
  master tcp ether *Server1-host* 5000
  query tcp ether *Server1-host* 5000
  hafailover *Server2*

Server2
  master tcp ether *Server2-host* 5001
  query tcp ether *Server2-host* 5001
  hafailover *Server1*

On Windows platforms, set these values in the*%JAGUAR%\ini\sql.ini* file:

[*Server1*]
master=NLWNSCK,*Server1-host*,5200
query=NLWNSCK,*Server1-host*,5200
hafailover=Server2

[*Server*2]
master=NLWNSCK,*Server2-host*,5300
query=NLWNSCK,*Server2-host*,5300
hafailover=*Server1*

Selecting the high availability option

To enable the high availability option for Client Library 11.0 connections, edit the connection cache properties using EAServer Manager.

❖ **Enabling high availability**

1 Expand the Connection Caches folder.

2 Highlight the connection cache for which you want to enable high availability.

3 Select File | Properties.

4 On the Driver tab, select Use HA Connection.

Using the connection APIs

Set the CS_HAFAILOVER property using the ct_config and ct_con_props CTLIB API calls. You can set this property at either the context or the connection level, using this syntax:

```
ct_config(context, action, CS_HAFAILOVER, buf, buflen, outlen)
```

```
ct_con_props(connection, action, CS_HAFAILOVER, buf, buflen, outlen)
```

For more information on using the CTLIB API calls, see *Using Sybase Failover in a High Availability System*, which is available in the Sybase online book collection at http://manuals.sybase.com/onlinebooks/group-as/asg1250e/hi_avail/@Generic__BookView.

CHAPTER 8

Setting up the Message Service

This chapter describes how to configure your server to use the message service.

Configuring the message service

Before you can use the message service to send and receive messages, you must configure the service and install the message service parts. Once you configure the message service, it is available to every server that you create.

EAServer includes an ASA database, which is used to store the message service configuration information. To use an Oracle database instead:

1 Change to the EAServer *Repository/Component/CtsComponents* directory.

2 Copy *MessageServiceConfig.props.oracle* to *MessageServiceConfig.props*.

3 Follow the instructions in "Using the Configure Message Service wizard."

❖ Using the Configure Message Service wizard

You can enable the message service and configure the connection cache properties using the Configure Message Service wizard.

1 In EAServer Manager, highlight the Message Service folder, and select File | Configure Message Service.

Or, if you double-click the Message Service folder, a message box asks if you want to configure the message service. To start the wizard, select Yes.

2 Select the server for which you want to configure the message service, and choose Next.

Note The wizard adds "CtsComponents/MessageService" to the com.sybase.jaguar.server.services property in the */Repository/Server/<server_name>.props* file. Once you have configured the message service, you can enable it for another server either by adding this value to the server's *.props* file, or by using the wizard.

---

3 Enter a connection cache name or select one from the list, and click Next.

By default, the message service uses the preconfigured JavaCache. If you change the message service cache, set it to the name of a JDBC connection cache that allows cache-by-name access.

If you configure the message service to use JavaCache, start the jagdemo database using either *start_sampledb* (UNIX) or *jagdemo.bat* (Windows); both are located in the EAServer *bin* subdirectory.

Note Do not use sun.jdbc.odbc.JdbcOdbcDriver as the database driver for your message service connection cache. This driver does not work correctly when accessing binary column data.

If you are using a Sybase ASA or Adaptive Server Enterprise database, set these values:

- Database Driver – `com.sybase.jdbc2.jdbc.SybDriver`
- Server Name – `jdbc:sybase:Tds:`***`host`***`:2638`

---

4 Optionally, edit the connection cache properties, and test the connection using ping. You must have a working connection to use the message service.

5 Select Finish. A message box asks if you want to restart the server. To enable the message service, select Yes.

❖ **Configuring message service cluster, database, and debugging options**

1 If your server is in a cluster, each server in the cluster must share the same database for the message service data and metadata. For information about synchronizing a cluster, see Chapter 6, "Clusters and Synchronization."

2 Use a text editor to make these changes to the *MessageServiceConfig.props* file, located in the EAServer *Repository/Component/CtsComponents* directory:

- To enable message service debugging at server start-up, change the value of cms.debug from false to true.
- If your database server does not accept the default syntax, change the SQL commands in the *XX.YYYYYY* statements. If the database does not have an image type, change image to unbounded (large) binary. If the database driver cannot handle character set conversions, change the datatype of the varchar columns in the SQL schemas to varbinary.
- To limit the number of persistent messages in the <system> queue and other queues that do not have a maximum size configured, modify or add the default.maximum property, for example:

  ```
  default.maximum=100
  ```

  The default is 100. This setting restricts in-memory caching of persistent messages; you can change it to tune the memory used to hold persistent messages in the <system> queue and user-defined queues where the maximum property is 0 or a negative value.

  This setting also determines how many persistent messages EAServer reads into memory during message service initialization. A large default size can delay server start-up when there is a large backlog of unprocessed messages, since the message service reads this many messages into memory when initializing.

  The setting does not restrict the number of transient messages in the <system> queue or in user-defined queues where the maximum property is 0 or a negative value.

3 Restart EAServer. The message service starts automatically when you start the server.

Configuring the message service to use non-ASCII characters

To store, retrieve, and display non-ASCII text correctly when using the message service database, you must do one of the following:

- When you specify the Server Name for the message service connection cache, define the character set using a jConnect™ for JDBC™ URL of this form:

```
jdbc:sybase:Tds:<host>:<port>?charset=<charset>
```

- Change the character set of the message service database to be consistent with the language you are using:

  a Using a text editor, open the *MessageServiceConfig.props* file located in the EAServer */Repository/Component/CtsComponents* directory.

  b Change the datatype of the varchar columns in the SQL schemas to varbinary.

  c Drop any tables that contain a column whose datatype you changed from the message service database.

Adding and configuring the message service parts

Once you have configured the message service, you can perform all administrative tasks using the CtsComponents::MessageService CORBA API—see Chapter 31, "Using the Message Service," in the *EAServer Programmer's Guide* for more information. You can also use EAServer Manager to add and configure the message service parts:

- Permanent destinations – add one message queue for each message recipient. To identify the subject of messages to which you want to subscribe, add message topics. You can also create message queues and topics using the CORBA API or the Java Message Service (JMS) createQueue and createTopic methods. Although these JMS methods are not portable, creating message queues and topics programmatically can significantly reduce the system administrator's work.
- Connection factories – queue connection factories enable JMS client applications to establish point-to-point (PTP) connections with the message service. Topic connection factories enable JMS client applications to establish publish/subscribe (Pub/Sub) connections with the message service.
- Message selectors – to specify which messages you want to receive, add message selectors to message queues.
- Message listeners – to provide asynchronous message notification for clients and components, implement a message listener, and install it on a message queue.

- Access roles – to restrict access to a message queue or to grant one user access to another user's queue, add access roles to the message queue. To restrict access to messages with particular topics, add access roles to the topic.
- Thread pools – define thread pools to handle client and component notification.

Permanent destinations

To provide permanent destinations for JMS client applications, define message queues and message topics using EAServer Manager. When you create a permanent destination, you can optimize its configuration properties, which benefits every JMS client application that uses the destination. Message queues and topics are both called message consumers.

❖ **Adding a message queue**

1 Highlight Configured Queues, right-click, and select Add.

2 In the New Queue dialog box, enter a name for the message queue.

3 Click OK.

❖ **Configuring a message queue's properties**

1 To edit properties for an active queue, select Active Queues; to edit properties for an inactive queue, select Configured Queues.

2 Select a queue from the list, right-click, and select Properties.

3 In the Queue Properties dialog box, select the Configuration tab.

4 To edit a configuration property, click on its property value. The Modify Property dialog box opens.

5 Edit the property value and click OK. The properties are defined in Table 8-1.

Table 8-1: Message queue properties

Property	Datatype	Default value	Description
IGNORE_DUPLICATE_KEY	boolean	false	Set to true to avoid duplicate messages when the message producer sends a message outside a transaction or forwards a message from a remote system. You can use this option with the CORBA API, where you can specify the message key before producing a message.
maximum	long	0	The maximum number of messages held in memory for an active queue. When the limit is reached, messages are discarded in the order that they would have been retrieved. The default of 0, or a negative size value, specifies that there is no limit on the number of transient messages in the queue. In this case, the number of persistent messages in the queue is limited by the default.maximum setting in the *MessageServiceConfig.props* file—see “Configuring the message service” on page 161. If a persistent message is discarded from memory, you can still retrieve it from the database. Transient messages are not stored in the database, so if they are discarded from memory, they are lost.
qop	string	“none”	Indicates the quality of protection required for the message queue object.
REQUIRES_ACKNOWLEDGE	boolean	false	Set to true to guarantee that each persistent message is delivered at least once. Transient messages may be lost if a server fails.
REQUIRES_TRANSACTION	boolean	false	Set to true to guarantee that a persistent message is delivered only once, which means that only one transaction can receive the message and successfully commit.
share	boolean	true	Indicates whether multiple clients can simultaneously receive messages from the queue. When a queue is not shared, only one client at a time can receive messages from it. If other clients try to receive a message, they get an OBJECT_NOT_EXIST system exception.
store	boolean	true	Indicates whether to store replicated transient messages that are added to the queue. To avoid duplicate message processing within a cluster where a shared queue may reside in memory on multiple servers, set to true.
table	string	“none”	To save the queue’s messages in a database table other than message_queue, enter the table name.

Property	Datatype	Default value	Description
timeout	long	0	The number of seconds the queue remains in memory when it is not being accessed by a client and it has no registered listener; set to zero or a negative number for no timeout. Any transient messages that are in memory when a timeout occurs are discarded.

❖ **Deactivating a message queue**

1 Select Active Queues.

2 Select a queue from the list, right-click, and select Close.

❖ **Deleting all the messages in a queue**

1 Select Active Queues.

2 Select a queue from the list, right-click, and select Flush.

❖ **Deleting a message queue**

1 Select Configured Queues.

2 Remove any listeners attached to the queue as follows:

a Highlight the message queue, and select File | Properties.

b In the Properties dialog box, select the Listeners tab.

c Highlight the listener name and click Delete.

3 Right-click the queue and select Delete.

❖ **Adding a topic**

1 Highlight Configured Topics, right-click, and select Add.

2 In the New Topic dialog box, enter the topic name.

3 Click OK.

❖ **Configuring a topic**

1 To configure an active topic, select Active Topics; to configure an inactive topic, select Configured Topics.

2 Select a topic from the list, right-click, and select Properties.

3 In the Topic Properties dialog box, select the Configuration tab.

4 To edit the timeout property, click on the property value. A Modify Property dialog box opens.

5 Edit the timeout value and click OK.

Property	Datatype	Default value	Description
timeout	long	0	Indicates the number of seconds the topic remains in memory when no active queues have selectors registered for the topic; set to zero or a negative number for no time out. Topics with no timeout are activated at server start-up.

❖ **Deleting a topic**

1 Select Configured Topics.

2 Select a topic from the list, right-click, and select Delete.

Note The message service includes a message queue and a thread pool called "<system>" for tasks that require internal messaging, such as synchronizing a cluster. The <system> message queue and thread pool are visible in EAServer Manager and as output from some jagtool commands. You cannot change the configuration of the <system> thread pool. You can change some properties of the <system> message queue, but only by changing the settings in the *MessageServiceConfig.props* file—see "Configuring the message service" on page 161.

Connection factories

To enable JMS applications to establish connections with the message service, create queue connection factories for PTP messaging, and create topic connection factories for Pub/Sub messaging.

❖ **Adding a queue or topic connection factory**

1 To add a queue connection factory, select Queue Connection Factory; to add a topic connection factory, select Topic Connection Factory.

2 Right-click and select Add.

3 Enter a name for the connection factory and click OK.

❖ **Configuring a connection factory's properties**

1 Select either Queue Connection Factory or Topic Connection Factory.

2 Highlight the connection factory you want to configure, right-click, and select Properties.

3 Select the Configuration tab and enter the connection factory properties described in Table 8-2.

Table 8-2: Connection factory properties

Property	Datatype	Default value	Description
CLIENT_ID	string	blank	Used for topic connection factories only. It enables EAServer to uniquely identify a client if it disconnects and later reconnects. Every connection that is created using this connection factory inherits this ID. The CLIENT_ID is ignored for queue connections.
CONFIG_QUEUE	string	blank	When you access a queue or topic, its configuration properties are copied from this named queue.
IGNORE_DUPLICATE_KEY	boolean	true	Indicates whether a message should be ignored if it is a duplicate.
NO_IMPLICIT_CREATION	boolean	true	Indicates whether a queue or topic that does not already exist should be implicitly created when a client attempts to access it.
REQUIRES_ACKNOWLEDGE	boolean	false	To optimize the performance of a JMS application, set to false. In this case, the message service does not acknowledge messages. If a connection terminates unexpectedly, messages may be lost.
REQUIRES_TRANSACTION	boolean	true	To force the methods publish, send, receive, and onMessage to participate in transactions, set to true. To improve throughput for bulk publishing, sending, or receiving transient messages, set to false. For information about using transactions, see Chapter 2, “Understanding Transactions and Component Lifecycles,” in the *EAServer Programmer’s Guide*.

Property	Datatype	Default value	Description
SHARED_LISTENER	boolean	false	When set to true, all message consumers for a connection use the same message listener. A shared listener can greatly improve performance for nondurable topic subscribers by creating a single message queue for all the topic subscriptions. To use this feature, install a message listener on the first topic subscription, then each nondurable subscription that uses the connection, receives messages from this listener. EAServer imposes two restrictions for shared listeners: • Do not call setMessageListener with a null parameter. This shuts down the current listener, which may be in use by other subscribers. • Do not call setMessageListener with the name of another listener, which shuts down the current listener and register the new listener.
SUPPORTS_TRANSACTION	boolean	true	Determines whether JMS publish, send, receive, and onMessage calls use the SUPPORTS_TRANSACTION option. If you set this to false, it significantly improves throughput for transient message bulk processing.
THREAD_POOL	string	blank	The message service uses threads from this thread pool for client notification. Using a thread pool can significantly improve performance.
TRANSPARENT_FAILOVER	boolean	true	Indicates whether JMS clients should allow transparent failover for message service operations. If set to false, a **pinned object** is used. If set to true, carefully consider the IGNORE_DUPLICATE_KEY setting.

Message selectors

To filter the messages you receive and to subscribe to specific message topics, add message selectors. You can add as many selectors as you want to each message queue.

❖ **Adding a message selector**

1 If you are adding a selector to an active queue, select Active Queues; if you are adding a selector to an inactive queue, select Configured Queues.

2 Select a message queue from the list, and select File | Properties.

3 In the Queue Properties dialog box, choose the Selectors tab.

4 Click Add.

5 In the New Selector dialog box, enter the selector. For example, to receive all published messages with the topic "StockPrice.SY", add this selector to the message queue:

```
StockPrice.SY
```

A selector is an expression that contains an equality condition for a topic and possibly other conditions. Selectors must conform to the JMS selector specification, which is a subset of the SQL-92 syntax. See Chapter 31, "Using the Message Service," in the *EAServer Programmer's Guide* for more information.

6 Click OK.

❖ **Deleting a message selector**

1 If you are deleting a selector from an active queue, select Active Queues; if you are deleting a selector from an inactive queue, select Configured Queues.

2 Select a message queue from the list, and select File | Properties.

3 In the Properties dialog box, click on the Selectors tab.

4 Choose a selector from the list and click Delete.

Listeners

To provide asynchronous message notification, implement and install a message listener on a message queue. A message listener can be either:

- An EJB 2.0 message-driven bean (MDB) that implements the javax.jms.MessageListener interface, or
- An EAServer class that implements the CtsComponents::MessageListener interface; this is called an EAServer message listener.

For information on how to implement, install, and configure an MDB, see Chapter 31, "Using the Message Service," in the *EAServer Programmer's Guide*.

❖ **Installing an EAServer message listener**

1 To install a listener on an active queue, select Active Queues; to install a listener on an inactive queue, select Configured Queues.

2 Select a message queue from the list, and select File | Properties.

3 In the Queue Properties dialog box, select the Listeners tab.

4 Click Add.

5 In the New Listener dialog box, enter a listener (component) that has been installed in EAServer. Use this format:

```
package_name/component_name[threadpool_name]
```

where *package_name* is the package name, *component_name* is the component name, and *threadpool_name* is an optional thread pool name. You can create thread pools in EAServer Manager as described in "Thread pools" on page 174. The thread pool must have one or more worker threads.

Note A thread pool with multiple worker threads enables the message listener to process multiple messages at the same time.

If you do not specify the name of a thread pool, the message listener uses the <system> default thread pool, which has a single worker thread.

6 Click OK.

Before you install the message listener component, configure it to support the MessageListener interface.

❖ **Configuring a component to support the CtsComponents::MessageListener interface**

1 Select EAServer | Installed Packages | *package_name* | *component_name*, where *package_name/component_name* is the name of an installed message listener.

2 Under the component, select Interfaces. Right-click and select Add Interfaces. This displays the Install Interfaces dialog box.

3 In the Available IDL Interfaces drop-down list, select CtsComponents. This displays the list of CtsComponents interfaces.

4 Select CtsComponents::MessageListener, and click Add. This moves the interface name to the Selected to Install list.

5 Click Install.

Alternately, you can use the IDL editor and modify your component's existing interface to inherit from CtsComponents::MessageListener; for example:

```
module msglistener
```

```
{
  interface Receiver : ::CtsComponents::MessageListener
  {
    ...
  }
};
```

❖ **Deleting an EAServer message listener**

1 To delete a listener from a message queue, select either Active Queues or Configured Queues.

2 Select a message queue from the list, and select File | Properties.

3 In the Properties dialog box, select the Listeners tab.

4 Choose a listener from the list and click Delete.

Access roles

To control access to message queues and topics, add one or more access roles. If more than one access role is assigned to a message queue or topic, a client must possess only one of the roles to access the queue or topic. If a message queue or topic has no assigned roles, any client can access it.

❖ **Adding an access role**

1 To add an access role to a message consumer, select the appropriate folder:

Message consumer	Folder title
Active queue	Active Queues
Inactive queue	Configured Queues
Active topic	Active Topics
Inactive topic	Configured Topics

2 Select a queue or topic from the list, and select File | Add.

3 In the Properties dialog, select the Roles tab.

4 Click Add.

5 In the New Role dialog box, enter the role name. You can use a wildcard character (“*”) when the queue or topic does not require a specific role. For subtopics that do not require a specific role, the name can end with the wildcard character; for example “StockPrice.*”.

6 Select the role type:

- **Consumer** To receive messages from the queue, or to receive messages with the specified topic.
- **Producer** To publish or send messages to the queue, or to publish or send messages with the specified topic.
- **Security** To administer roles for the message queue or topic.

7 Click OK.

❖ **Deleting an access role from a message queue or topic**

1 To delete an access role from a message consumer, select the appropriate folder:

Message consumer	Folder title
Active queue	Active Queues
Inactive queue	Configured Queues
Active topic	Active Topics
Inactive topic	Configured Topics

2 Select a message queue or topic from the list, and select File | Properties.

3 In the Properties dialog box, select the Role tab.

4 Select a role from the list and click Delete.

Thread pools

To provide asynchronous client and component notification, define thread pools and specify the number of threads dedicated to each type of notification.

❖ **Adding a thread pool**

1 Select Thread Pools, right-click, and select Add.

2 In the New Thread Pool dialog box, enter the name of the thread pool.

3 Click OK.

Reader, writer, and worker threads To use a thread pool for client notification, set the value of readers to "3", writers to "2", and workers to "0". Based on your own performance measurements, increase the number of reader and writer threads if it improves throughput. Using thread pools to improve performance is generally suitable only for high-volume client notification with transient messages. When message delivery is transactional or IIOP/SSL via the QOP property, the thread pool's reader and writer threads are not used.

To use a thread pool for component notification, set the values of both readers and writers to "0". Set the value of workers to "1" unless you want to allow parallel message processing, in which case you would increase this value.

❖ **Modifying the number of threads in a thread pool**

1 Select Thread Pools.

2 Select a thread pool from the list, and select File | Properties.

3 In the Thread Pool Properties dialog box, select the Configuration tab.

4 To edit the number of threads, click on one of the property values.

Property	Datatype	Default value	Description
readers	long	0	The number of reader threads in the thread pool, which are used for client notification.
writers	long	0	The number of writer threads in the thread pool, which are used for client notification.
workers	long	0	The number of worker threads in the thread pool, which are used for component notification.

5 Modify the property value and click OK.

Multiple MDB instances

To enable EAServer to create multiple instances of a message-driven bean (MDB):

1 Create a thread pool for component notification, and set the workers property to a value greater than 1.

2 Assign this thread pool to the MDB:

a In EAServer Manager, highlight the MDB, and select File | Properties.

b On the MDB Type tab, append the name of the thread pool you just created to the Listener name. For example, if you created a thread pool called “threads1” and the Listener Name is *MyPkg/MyComp*, change the Listener Name to `MyPkg/MyComp[threadsl]`.

See Chapter 31, “Using the Message Service,” in the *EAServer Programmer’s Guide* for more information about configuring MDBs.

❖ **Deleting a thread pool**

1 Select Thread Pools.

2 Select a thread pool from the list, right-click, and select Delete.

Dead queue

If the message service cannot deliver a message, it moves the message to the dead message queue. You can view a list of these messages, delete them, or resend them to their original destination.

❖ **Viewing the dead message queue**

1 Select Dead Queues.

2 To see a list of the dead messages, select Dead Queue from the list, right-click, and select List.

To resend a dead message, select a message from the list and click Send.

3 To delete the dead messages, select Dead Queue from the list, right-click, and select Delete All Dead Messages.

Viewing messages and statistics

❖ **Viewing a list of the messages in a queue**

1 Select Active Queues.

2 Select a message queue from the list, right-click, and select List.

❖ **Viewing statistics for all message queues and topics**

- Select Statistics.

❖ **Viewing a message queue's statistics**

1. To view statistics for an active queue, select Active Queues; to view statistics for an inactive queue, select Configured Queues.
2. Select a message queue from the list, and select File | Properties.
3. In the Active Queue or Config Queue Properties dialog box, select the Statistics tab.

❖ **Viewing a message topic's statistics**

1. Select Configured Topics.
2. Select a topic from the list, and select File | Properties.
3. In the Topic Properties dialog box, select the Statistics tab.

CHAPTER 9

Importing and Exporting Application Components

This chapter describes how to deploy application components to EAServer using standard J2EE archive formats and EAServer JAR format, including the import and export of archive files for:

- Packages
- Components
- Web applications
- J2EE applications
- J2EE application clients
- J2EE connectors

The J2EE archive formats allow you to interchange application components between J2EE servers from different vendors. The EAServer JAR format allows you to more easily copy application components between different EAServer installations.

Deploying packages and components

Components must be archived as part of the package where they are installed. You can archive in two formats:

- **EAServer JAR** This format, while proprietary to EAServer, supports all component types. Also, unlike the EJB-JAR format, all component property settings are preserved in the archive.
- **EJB JAR** This format provides portability between J2EE and EJB servers from different vendors. Only EJB components are archived in this format; components of other types are ignored when you create the archive. Not all component properties are preserved in the archive, because the deployment descriptor does not support all EAServer properties. These EAServer properties can be configured by including an EAServer XML configuration file in the archive, as described in "Using EAServer configuration files in J2EE archives" on page 196. The following settings in the Component Properties dialog box are not configured by the EJB-JAR deployment descriptor, and must be configured in the XML configuration file or manually after deployment:
  - For beans that use container-managed persistence, the Persistence settings.
  - Role mappings and method-level permissions.
  - Resource references.
  - EJB references to components that are not installed with the JAR file or when multiple beans use the same home and remote interfaces. It is impossible to infer EJB references if more than one bean uses the home and remote interfaces specified by the reference properties in the deployment descriptor. After importing an EJB-JAR file that contains multiple beans that use the same home and remote interfaces, view the EJB Reference properties to verify that the correct bean is invoked.
  - Environment properties.
  - For EJB 2.0 components, Resource Environment Refs properties.
  - For EJB 2.0 components, Run As Identity properties.

  For information on these properties, see the EAServer Manager online help or the *EAServer Programmer's Guide*.

- **EJB 1.0 JAR** This format supported for backward compatibility. It is similar to the EJB-JAR format, but supports only EJB version 1.0 components. For more information on this format, see Appendix A, “Using EJB 1.0 JAR Support”

Importing and exporting components in EAServer JAR format

EAServer Manager allows you to create an archive file containing component definitions and implementation files for all components in a package or a single component. You can export archive files from packages on your development server and import them to your production server. If the JAR file contains a single component, it must be installed to a package with the same name as the one from which it was exported.

❖ **Deploying packages between servers using archive files**

1 Start EAServer Manager on the test server host and connect to the test server.

2 Optionally, configure the package properties to specify the list of additional files to be included in the package archive. In the Package Properties window, you can use the Additional Files tab to set the com.sybase.jaguar.package.files property, as described in the online help.

   By default, all component implementation files and required stub files are included in the archive. You may want to include other files, such as project descriptions, client applets, and HTML pages. You can specify additional files either by setting the package or component properties or when generating the package archive.

3 Export an archive of the application’s EAServer packages. This step creates a Java archive (JAR) file containing the component definition files and implementation files. “Exporting a package archive” on page 182 describes this step in detail.

4 Copy the JAR file to the production server.

5 Start EAServer Manager on the production server host, and connect to the production server.

6 Import the JAR file containing the package archive into the production server. “Importing a package archive” on page 182 describes this step in detail.

7 If you have copied the package between host machines that have different architectures, recompile the components to run on the new architecture.

❖ **Exporting a package archive**

1. Start EAServer Manager, and connect to the server where the package is installed.
2. Highlight the package you are exporting.
3. Select File | Export | EAServer JAR.
4. The Export dialog box appears.

   For each C/C++ component in your package, provide the name of the corresponding DLL or shared-library file. No input is required for Java components.
5. Click the Misc. Information button to add any other files that you want to include with the package. These files can include HTML files, project files, and so on.

   Use the Browse, Add, and OK buttons to add other files to the JAR file. To delete a miscellaneous file, highlight the file and click Delete. Click Done when all the files you want to include are listed in the dialog box.
6. Click OK.

The exporter locates the Java class and C/C++ shared library files and combines them with the miscellaneous files you selected into a JAR file.

❖ **Importing a package archive**

1. Copy the JAR file containing the package definition to the host machine for the target server.
2. Start EAServer Manager and connect to the target server.
3. Expand the Packages folder and verify that the package to be imported does not already exist. If it does, select it and delete the package.
4. Highlight the packages folder, and select File | Import.
5. The Import dialog box appears:
   - Use the Browse button to locate the JAR file that you are importing.
   - Enter the full path of the directory where you want the archive to be unbundled. This directory becomes the root directory from which the JAR file is unbundled. Unbundling creates the subdirectories, class files, DLLs, and any other files that were included in the exported JAR file.
6. Click Import.

❖ **Exporting a component as an EAServer JAR file**

1 Highlight the icon for the component you are exporting. .

2 Select File | Export | EAServer JAR.

3 The Export dialog box appears. Enter the full path of the directory where the JAR file is to be created.

4 Click OK. EAServer Manager creates the file name *component.jar*, where *component* is the name of the selected component.

❖ **Importing a single component from an EAServer JAR file**

If the JAR file contains a single component, it must be installed in a package with the same name as the one where it was originally installed. Create this package if necessary. Import the component as follows:

1 Highlight the package from which the component was exported.

2 Choose File | Deploy | EAServer JAR.

3 Use the Browse button to select the JAR file, or type the full path to the file.

4 Click OK to begin importing.

Importing and exporting packages in EJB-JAR format

An EJB-JAR file contains the implementation classes, interface classes, and deployment descriptor for one or more beans. You can use a Java development tool such as Sybase PowerJ to define and develop beans and create an EJB-JAR file. You can import JAR files in the EJB 1.0, EJB 1.1, or EJB 2.0 formats. EAServer Manager reads the JAR file and creates a package containing a component for each bean in the JAR file.

PowerJ deploys Enterprise JavaBeans directly to EAServer If you are developing in PowerJ, use the Enterprise JavaBeans Deployment Wizard to install EJB components to EAServer. If using another IDE, use EAServer Manager to import the bean as described below.

❖ **Importing an EJB 1.1 or 2.0 JAR file**

1 Start EAServer Manager if it is not already running, and connect to the server where you want to install the component.

2 Highlight the top-level Packages folder. Choose File | Deploy | EJB JAR.

3 Enter the path to the EJB-JAR file.

4 Choose the Deployment Strategy from the options described in Table 9-1.

Table 9-1: Deployment strategy options

Deployment strategy	Specifies
Full deployment	The importer generates IDL for every class defined in the JAR, regardless of whether the interface already exists. Use this option when deploying components for the first time or when you want to restore IDL types that have been changed or deleted.
Incremental deployment	The importer generates IDL only when the Java types and interfaces have changed from the last time it was imported. The following are compared: • Methods • Fields • Interfaces • Superclass Use this option if you redeploying components and have changed some interfaces or parameter types.
Optimistic deployment	Similar to Incremental, except that the check for changed classes in each package ends if the first class comparison indicates no change. Use this option if you are redeploying components, and have changed only the implementation classes.

5 Configure the Deployment Options:

- **Prompt before overwriting existing objects** If selected, the importer will confirm any overwrites of existing components, IDL definitions, and other repository entities that may be redefined by the import process.
- **Automatically generate EJB stubs and skeletons** Select this option if you want the importer to generate and compile stubs and skeletons for the new components. To allow compilation, the classes in the EJB-JAR file must be self-contained, in other words classes in the JAR file cannot depend on classes that are not in the JAR file or part of the standard J2EE class distribution.

Home interfaces that use java.util.Enumeration
Do not select the Automatically generate EJB stubs and skeletons option if entity bean finder methods return java.util.Enumeration. Instead, generate Stubs and Skeletons after the import completes, and select the JDK 1.3 or later option.

All entity bean finder methods within one EJB JAR file must return the same list type, either java.util.Collection or java.util.Enumeration.

- **Use interoperable naming** If selected, naming URLs in the EJB Reference properties will use interoperable naming URLs, as described in "Intervendor EJB interoperability" in the *EAServer Programmer's Guide*. Select this option when your EJBs have EJB references that link to another vendor's EJB 2.0 server and you need to use the RMI/IIOP protocol for the connection.
- **Use JAR File Package Naming Naming** If selected, the new package name will match the EJB-JAR file name. Otherwise the new package name matches the `display-name` element in the deployment descriptor.

Note When you are deploying an EAR or EJB-JAR file containing a *sybase-easerver-config.xml* file, which was previously exported from EAServer, do not use the JAR File Naming option. When exporting packages to a J2EE archive file, EAServer optionally creates a *sybase-easerver-config.xml* file that includes configuration information based on the package names. If you deploy a package or an application that contains packages with the JAR File Naming option, EAServer renames the packages. When the package names do not match the names in the *sybase-easerver-config.xml* file, deployment fails.

6 Click Finish.

EAServer Manager creates a new package that contains a component for each bean defined in the JAR file, printing status messages and warnings to the Deploy Wizard. The new package has the same name as the EJB JAR display name. If there is no display name, the new package has the same name as the JAR file. For each bean in the EJB-JAR, EAServer creates an EJB component with the same name as the ejb-name element in the EJB-JAR deployment descriptor.

Home names for imported EJB components EAServer sets an imported bean's home name to the EAServer default, *package/component*, where *package* is the EAServer Manager package name, and *component* is the EAServer Manager component name.

Use the status dialog as a to-do list In the deployment status dialog box, EAServer Manager display warnings for each setting that requires further attention before running the application. You can copy and paste this text to a text editor to use as a to-do list.

Exporting EJB-JAR files

You can create an EJB-JAR file that contains the Java classes and deployment descriptors for the EJB components installed in an EAServer package. The JAR file can be deployed to another EAServer installation or any EJB compatible server.

You can export EJB-JAR files in two formats:

- **EJB JAR** Use this format for exporting EJB 2.0 or 1.1 components. Components in the package of other types are not included in the JAR file. The file can be deployed to any EJB 2.0 server.
- **EJB 1.0 JAR** Use this format for exporting EJB 1.0 components. Components in the package of other types are not included in the JAR file. The file can be deployed to any EJB 1.0 server. Appendix A, "Using EJB 1.0 JAR Support," describes this option.

Use synchronization for deploying beans between servers The EAServer synchronization feature can also be used to deploy components from one EAServer installation to another. Synchronization is simpler than importing and exporting JAR files, but the source server must be able to connect to the target server. If you use synchronization, you will avoid the need to reconfigure JNDI resource references on the target server for EJB 2.0 or 1.1 components.

❖ **Exporting an EJB-JAR file**

1 Highlight the EAServer package to export and choose File | Export, then choose EJB 1.0 JAR or EJB JAR.

2 Enter the path and file name for the new JAR file and click Next.

3 Optionally deselect the Export with EAServer XML Configuration File option if you do not want the archive to include an EAServer XML configuration file.

4 EAServer Manager creates the JAR file, displaying status messages in the Export wizard.

Deploying Web applications

You can export Web applications from EAServer Manager to deploy them on another server. EAServer supports two archive formats for Web applications:

- **J2EE Web archive (WAR)** The WAR format is the standard for servers that support J2EE. This format allows portability to other vendor's J2EE servers, but not all properties are preserved in the archive, because the deployment descriptor does not support all EAServer properties. These EAServer properties can be configured by including an EAServer XML configuration file in the archive, as described in "Using EAServer configuration files in J2EE archives" on page 196. In particular, these items are not supported by the WAR file deployment descriptor:
  - Resource references, described in the "Creating Web Applications" chapter in the *EAServer Programmer's Guide*
  - EJB references, described in the "Creating Web Applications" chapter in the *EAServer Programmer's Guide*
  - Environment properties, described in the "Creating Web Applications" chapter in the *EAServer Programmer's Guide*
  - Resource environment references, described in the "Creating Web Applications" chapter in the *EAServer Programmer's Guide*
  - Security access role mappings, described in the *EAServer Security Administration and Programming Guide*
- **EAServer JAR** For exporting between EAServer 3.6 or later servers. This format, while proprietary, preserves all information in the Web application. When importing, EJB references, resource references, and role mappings are preserved. You must ensure that the referenced items are in place before running the imported Web application.

Consider the synchronize feature instead of using archives If you are deploying between EAServer installations, you may find the synchronize feature easier than exporting and importing archives. Synchronization replicates a Web application directly between servers. See Chapter 6, "Clusters and Synchronization" for more information.

❖ **Exporting a Web application**

1 Expand the Web Applications folder, then highlight the icon that represents your application.

2 If you are exporting in WAR format, choose File | Export | J2EE WAR. If you are exporting in EAServer JAR format, choose File | Export | EAServer JAR.

3 Enter a path and file name for the file to be created, including the *.war* or *.ear* extension.

4 If you are exporting in WAR format, optionally deselect the Export with EAServer XML Configuration File option if you do not want the archive to include an EAServer XML configuration file.

5 Click Next. The Export wizard creates an archive of your Web application, displaying status information in the window.

6 When the export is complete, click Close.

❖ **Importing a Web application**

1 Highlight the top-level Web Applications folder. If importing a WAR file, choose File | Deploy | J2EE WAR. If importing an EAServer JAR, choose File | Deploy | EAServer JAR.

2 Enter the path to the WAR or JAR file.

3 Optionally, check to enable:

- Prompt before overwriting existing objects
- Automatically generate EJB stubs and skeletons
- Use interoperable naming, which configures interoperable naming URLs for the EJB Reference properties, as described in "Intervendor EJB interoperability" in the *EAServer Programmer's Guide*. Select this option when your EJBs have EJB references that link to another vendor's EJB 2.0 server and you need to use the RMI/IIOP protocol for the connection.

4 Click Next. The Deploy wizard reads the file and creates the Web application. Any errors are displayed in the status window. Review the status information, then click Close.

Use the status dialog as a to-do list In the deployment status dialog box, EAServer Manager displays warnings for each setting that requires further attention before you can run the Web application. You can copy and paste this text to a text editor to use as a to-do list.

What is created during import

When importing na EAServer JAR, the Deploy wizard creates a Web application that is identical to the original.

When importing a WAR, the Deploy wizard creates a Web application with the same name as the display name in the WAR file's XML descriptor. If there is no display name, the new Web application has the same name as the WAR file. For each servlet defined in the WAR, the Deploy wizard creates a Web component with the same name as the servlet-name element in the Web application deployment descriptor.

Before running servlets or JSPs in the Web application, you may need to configure the following settings in the Web Application Properties dialog box:

- Role mappings
- Resource references
- EJB references
- Environment properties

Deploying J2EE applications

You can export applications from EAServer Manager to deploy them on another server. EAServer supports two archive formats for applications:

- **J2EE enterprise archive (EAR)** The EAR format is the standard for servers that support J2EE. This format allows portability to other vendor's J2EE servers, but does not support component types other than EJB or container-specific information such as:

- Role mappings
- Resource references
- EJB references to components that are not installed with the EAR file, or when more than one bean uses the same home and remote interfaces. It is impossible to infer EJB references if more than one bean uses the home and remote interfaces specified by the reference properties in the deployment descriptor. After importing an EJB-JAR file that contains multiple beans that use the same home and remote interfaces, view the EJB Reference properties to verify that the correct bean is invoked.
- Environment properties

You can optionally include EAServer XML configuration files to preserve the configuration of these properties, as described in "Using EAServer configuration files in J2EE archives" on page 196.

- **EAServer JAR** For exporting between EAServer 3.6 or later servers. This format, while proprietary, preserves all information in the application and supports component types other than EJB. When importing, EJB references, resource references, and role mappings are preserved. You must ensure that the referenced items are in place before you run the imported application.

Consider synchronization instead of archives If you are deploying between EAServer installations, you may find synchronization easier than exporting and importing archives. Synchronization replicates an application directly between servers. See Chapter 6, "Clusters and Synchronization"for more information.

❖ **Exporting an application**

1 Expand the top-level Applications folder.

2 Highlight the application to export. If you are exporting an EAR file, choose File | Export | J2EE EAR. If you are exporting an EAServer JAR, choose File | Export | EAServer JAR.

3 In the Export wizard:

a Enter the name of the EAR or JAR file to create including the full directory path and the *.ear* or *.jar* extension.

b If exporting in EAR format, optionally deselect the Export with EAServer XML Configuration File option if you do not want the archive to include an EAServer XML configuration file.

4 Click Next. EAServer Manager creates the EAR or JAR file, displaying status messages in the Export wizard.

❖ **Importing an application**

1 Highlight the top-level Applications folder. If you are importing an EAR file, choose File | Deploy | J2EE EAR. Otherwise, choose File | Deploy | EAServer JAR.

2 Enter the path to the EAR or JAR file.

3 If importing an EAR file, choose a deployment strategy. Table 9-1 on page 184 describes the deployment strategy options.

4 If importing an EAR file, configure the deployment options:

- Prompt before overwriting existing objects.
- Automatically generate EJB stubs and skeletons.
- Use interoperable naming, which configures interoperable naming URLs for the EJB Reference properties, as described in "Intervendor EJB interoperability" in the *EAServer Programmer's Guide*. Select this option when your EJBs have EJB references that link to another vendor's EJB 2.0 server and you need to use the RMI/IIOP protocol for the connection.
- Use JAR File Package Naming. If selected, the new package name will match the EJB-JAR file name. Otherwise the new package name matches the `display-name` element in the deployment descriptor.

Note When you are deploying an EAR or EJB-JAR file containing a *sybase-easerver-config.xml* file, which was previously exported from EAServer, do not use the JAR File Naming option. When exporting packages to a J2EE archive file, EAServer optionally creates a *sybase-easerver-config.xml* file that includes configuration information based on the package names. If you deploy a package or an application that contains packages with the JAR File Naming option, EAServer renames the packages. When the package names do not match the names in the *sybase-easerver-config.xml* file, deployment fails.

5 Click Next. The Deploy wizard reads the EAR or JAR file and creates a new application. Any errors display in the status window. Review the status information, then click Close.

What is created during import

When you import an EAServer JAR, the Deploy wizard creates an application identical to the original.

When you import an EAR, the Deploy wizard creates:

- An application with the same name as the display name in the EAR file's XML descriptor. If there is no display name, the new application has the same name as the EAR file.
- For each EJB-JAR file in the EAR, a package with the same name as the EJB-JAR display name, or the name of the EJB-JAR file if there is no display name.
- For each bean in an EJB-JAR file, an EJB component with the same name as the ejb-name element in the EJB-JAR deployment descriptor.

Home names for imported EJB components When importing from EAR or EJB-JAR files, EAServer sets an imported bean's home name to the EAServer default, *package/component*, where *package* is the EAServer Manager package name, and *component* is the EAServer Manager component name.

- For each WAR file in the EAR, a Web application with the same name as the display name in the WAR file's XML descriptor. If there is no display name, the new Web application has the same name as the WAR file.
- For each servlet defined in a WAR file, a Web application component with the same name as the servlet-name element in the Web application deployment descriptor.

If the EAR file did not contain EAServer XML configuration files, you may need to configure the following settings in the Component or Web Application Properties dialog boxes before running EJBs, servlets, or JSPs:

- Role mappings
- Resource references
- EJB references (to components that are not installed with the EAR file)

- Environment properties
- Resource environment references

Other settings have been configured by the Deploy wizard.

Use the status dialog as a to-do list In the deployment status dialog box, EAServer Manager displays warnings for each setting that requires further attention before you run the application. You can copy and paste this text to a text editor to use as a to-do list.

Deploying application clients

You can export application clients from EAServer Manager to deploy them on a client machine. You can import application clients into EAServer as part of a J2EE EAR file or an EAServer JAR file. Typically, you may want to import the EAR file, edit the application client properties, export the JAR file, and deploy and run the application on a client machine.

For information on creating and running application clients, see Chapter 10, "Creating Application Clients," in the *EAServer Programmer's Guide*.

❖ **Importing an application client**

1 Highlight the top-level Applications folder.

2 To import a J2EE EAR file, choose File | Deploy | J2EE EAR. To import an EAServer JAR file, choose File | Deploy | EAServer JAR. This starts the Deploy wizard.

3 In the Deploy wizard, provide the following information:

- Enter the path and file name of the EAR or JAR file or click Browse and select the file from the system.
- Select a deployment strategy. Table 9-1 on page 184 describes the deployment strategy options.
- Deployment Options – check to enable:
  - Prompt before overwriting existing objects
  - Automatically generate EJB stubs and skeletons

- Use interoperable naming, which configures interoperable naming URLs for the EJB Reference properties, as described in "Intervendor EJB interoperability" in the *EAServer Programmer's Guide*. Select this option when your EJBs have EJB references that link to another vendor's EJB 2.0 server and you need to use the RMI/IIOP protocol for the connection.

4 Click Next. The Deploy wizard reads the file and creates the application client. Any errors are displayed in the status window. Review the status information, then click Close.

❖ **Exporting an application client**

1 Expand the icon that represents your application, open the Clients folder, and highlight the icon that represents your application client.

2 Choose File | Export Client JAR. This starts the Export wizard.

3 In the Export wizard:

a Enter a path for the JAR file to be created or click Browse and select the file path.

b Optionally deselect the Export with EAServer XML Configuration File option if you do not want the archive to include an EAServer XML configuration file.

4 Specify whether to automatically generate EJB stubs. To run the JAR file on a standalone client machine, generate the stubs. If you plan to deploy the JAR file in another application server, you do not need to generate the stubs.

5 Click Next to continue. EAServer exports the application client. Any errors are displayed in the status window. Review the status information, then click Close.

Note If you make changes to EJB references, resource references, or environment properties after you export the application client, you must export the client again to update the JAR file.

Deploying connectors

The J2EE connector architecture enables you to write portable Java applications that can access multiple transactional enterprise information systems. The EAServer connector implementation defines the relationship between the application server and a resource adapter, also known as a connector. A connector is a specialized connection factory that provides connections for EJBs, Java servlets, JSPs, and CORBA-Java components.

❖ Importing a connector

From EAServer Manager, you can import a connector from either an EAServer JAR file, or a J2EE resource archive (RAR) file.

1 Highlight Connectors.

2 Choose File | Deploy | [EAServer JAR | J2EE RAR]. For a JAR file, this displays the Specify JAR File dialog box. For an RAR file, it starts the Deploy wizard.

3 Enter the name of the JAR or RAR file, or click Browse and select the file.

4 In the Deploy wizard, select whether to have EAServer prompt you before it overwrites existing objects.

5 In the Specify JAR File dialog box, click OK. A message box informs you whether the file is successfully deployed.

In the Deploy wizard, click Next. The deploy wizard displays the deployment status.

❖ Exporting a connector

You can export a connector to either an EAServer JAR file or a J2EE RAR file.

1 Expand the Connectors folder, then highlight the connector you want to export.

2 Choose File | Export | [EAServer JAR | J2EE RAR]. For an EAServer JAR file, this displays the Export dialog box. For a J2EE RAR file, this displays the Export wizard.

3 Enter the location where you want EAServer to export the file, or click Browse and select the location. For an EAServer JAR file, click Export. A message box informs you whether the export is successful. For a J2EE RAR file, click Next. The Export wizard displays the export status.

Deploying other entity types

You can create entity collections to create archives for any entity type. For example, an entity collection can contain properties for a server, connection cache, packages, and Web applications. Entity collections must be archived using jagtool or the Repository API; EAServer Manager does not support entity collections. For more information, see:

- Entity collection properties on page 448
- Chapter 12, "Using jagtool and jagant." Entity collections can be imported and exported in EAServer JAR format, using the deploy and export commands.

Using EAServer configuration files in J2EE archives

You can embed XML configuration files in J2EE archives to configure EAServer properties that can not be specified by the standard deployment descriptor properties. "XML configuration files" on page 226 describes the format of these files.

XML files for EAServer must be named *sybase-easerver-config.xml* and placed in the *META-INF* directory. Whenever you import a J2EE JAR, WAR, EAR, or RAR file with EAServer Manager, jagtool, or jagant, the importer looks for this file and applies the changes specified within it.

When you export a J2EE archive, an XML configuration file is created with entries to configure the entities defined within the archive, as follows:

- For a Web application (WAR) archive, the XML file updates properties for the Web application, and any filters, servlets, and JSPs in the Web application.
- For EJB components in a package (exported as an EJB-JAR file), the XML file updates properties for the package and each EJB component inside it.
- For a J2EE connector (RAR) archive, the XML file updates properties for the connector, and creates any managed connection factories which exist for the Connector.
- For an application client archive, the XML file updates properties for the application client.

- For an application archive (EAR file), the XML file updates properties for the application. The EJB components, connectors, and application clients in the application are configured by separate XML files associated with the nested archives created for these entities.

To export an archive that does not contain an XML configuration file, deselect the "Export with EAServer XML Configuration File" option in the export wizard.

To create XML configuration files without exporting an archive, you can use the jagtool exportconfig command. See Chapter 12, "Using jagtool and jagant," for details.

Merging property files in EAServer JAR archives

You can add merge files to EAServer JAR files so that property files are merged rather than being overwritten. Normally, an EAServer JAR file contains all the properties for the entities exported in the archive. In some cases, you may not want to overwrite existing property files. For example, if you create an entity collection that contains a server, you may not want to overwrite the existing server properties when the archive is imported.

You can also add "cleaner" files, which cause entities to be removed entirely when an archive is exported.

❖ **Creating an archive that contains merge files**

1 Create the merge and cleaner files for the entities to be exported.

2 For each entity that requires a merge or cleaner file, add the file name to the entities .files property (that is, the property that ends in .files, such as com.sybase.jaguar.component.files).

3 Create the archive using jagtool or EAServer Manager.

Creating merge files

Merge files must exist in the same directory as the properties file to merge with, and have the same base name with the *.merge* extension. For example, to merge properties for server Jaguar, the JAR file must contain these files:

```
Server/Jaguar.props
```

```
Server/Jaguar.merge
```

If a property is set in the merge file and in the *.props* file, the value is merged based on the merge-file setting, and the *.props* file setting is ignored. If a property is not set by either the merge file or in the *.props* file, it is left as is when the archive is imported.

Merge files are text files, with one merge operation per line. Each merge operation uses the syntax:

```
MergeOperation:Property=NewValue
```

Where:

- *MergeOperation* is an operation listed in Table 9-2.
- *Property* is the property name.
- *NewValue* is the value to be merged with or removed from the existing value.

Table 9-2: Merge operations

Merge operation	Action
SetDefault	If the property is not set, or set with no value, set the property to the specified value. No action if the property is already set.
AppendToList	For properties that take comma-separated lists of values, append the specified value to the existing value and remove duplicate entries.
PrependToList	For properties that take comma-separated lists of values, prepend the specified value to the existing value and remove duplicate entries.
RemoveFromList	For properties that take comma-separated lists of values, remote the specified entry to the existing value.
Delete	Remove the property entirely.

Example merge operations

This example uses the AppendToList and RemoveFromList operations. For a server, this syntax could be used to remove an old version of a package and install a new version:

```
RemoveFromList:com.sybase.jaguar.server.packages=OEMPackageVersion1
AppendToList:com.sybase.jaguar.server.packages=OEMPackageVersion2
```

This example deletes the com.sybase.jaguar.server.timeout property:

```
Delete:com.sybase.jaguar.server.timeout
```

Creating cleaner files

If present in an EAServer JAR file, cleaner files cause an entity to be removed entirely when an JAR file is imported. Cleaner files must exist in the same directory as the properties file for the entity to be deleted, and have the same base name with the *.clean* extension. For example, to cause deletion of the component *MyStuff/Account*, the JAR file must contain these files:

```
Component/MyStuff/Account.props
Component/MyStuff/Account.clean
```

Deleting packages, Web applications, and applications

When deleting a package, Web application, or application, you can choose whether to delete only the entity's properties file, or to delete all files that have been generated by the deployment of the entity, including component stubs and skeletons and IDL interface and datatype definitions. By default, EAServer Manager performs a full deletion. If you choose a simple deletion, only properties files were deleted. When deleting with jagtool, add the type option to the delete command to specify whether to use full or simple deletion.

This feature is useful for deleting entities that have been imported from a J2EE archive file. Full deletion makes removing the entity as easily as the original deployment.

CHAPTER 10

Using Repository Versioning

Introduction

The EAServer configuration repository stores configuration and implementation files for installed application entities such as components, Web applications, JSPs, and Java servlets. Repository versioning allows you to save numbered versions of an entity. Each version archive contains configuration properties and implementation files associated with the entity. For example, before undertaking a new development phase, you might save a new major version of your J2EE application.

You can save major versions and minor versions. Major versions are indicated by a new starting number in the version; for example, 1.3 moves to 2.0. Minor versions are indicated by an increment of the second number; for example, 1.3 moves to 1.4.

For most entity types, a major version includes configuration information and implementation files. A minor version includes only implementation files.

You can restore any saved version. Restore replaces the current configuration and implementation files with the saved version files.

Creating and restoring versions

Using EAServer Manager, you can create versions for any repository entity type listed in Table 10-1.

Table 10-1: Version data for each entity type

Entity type	Version data contains
Server	For major and minor versions, includes only configuration properties for the server. **Warning!** Unlike the application entity type, the server version data does not include installed applications, packages, or Web applications. You must create versions of these entities yourself.
Application	Includes application properties data, plus version data for each Web application and package in the application. Major versions include package and Web application implementation files, as described below.
Package	Version data for the package plus version data for each component installed in the package. A major version includes: • Package property settings • Package files (the file set indicated by the com.sybase.jaguar.package.files property) • Major version data for each component in the package • For each component, implementation files (the file set indicated by the com.sybase.jaguar.component.files property plus the component implementation files). The major version file set is the same as would be included if the package were archived in Jaguar JAR format. A minor version includes only package properties and minor version data for each component in the package.
Component	Major and minor versions include only configuration properties.

Entity type	Version data contains
Web Application	A major version includes configuration properties and implementation files (the entire context root directory including the *WEB-INF* subdirectory), plus major version data for installed Web components. A minor version includes only configuration properties for the Web application and for installed Web components.
Servlet or Web Component	Major and minor versions include only configuration properties.
J2EE Client	A major version includes configuration properties and implementation files. A minor version includes only configuration properties.

❖ **Manually creating a new version**

1 Highlight the entity of interest.

2 Select File | Versioning, then select:

- Save Minor Version to create a new minor version, or
- Save Major Version to create a new major version.

3 Enter a comment describing the version and click OK.

4 EAServer Manager creates a new version archive for the entity. If the entity contains nested entities (see Table 10-1), EAServer Manager creates new versions for each nested entity.

❖ **Viewing version history**

Version history shows the version number, creation date, and comment for all available versions of an entity. To display version history:

1 Highlight the entity of interest and select File | Versioning | List Versions.

2 EAServer Manager displays a list of version numbers, creation dates, and the corresponding comments.

❖ Restoring a version

Restoring replaces the current configuration and implementation files with the saved version files.

Warning! Restore overwrites the current configuration and implementation. Save the current version first if you may need to restore it in the future.

To restore a version:

1 Highlight the entity of interest and select File | Versioning | Restore Version.

2 EAServer Manager displays a list of available versions and the corresponding comments.

3 Highlight the version to be restored, then click OK.

❖ Deleting a version

Deleting removes the version data and archived files so the version can no longer be restored. To delete a version:

1 Highlight the entity of interest and select File | Versioning | List Versions.

2 EAServer Manager displays a list of available versions and the corresponding comments in the List Versions dialog box. Highlight the version to delete, and click Delete.

CHAPTER 11

Runtime Monitoring

This chapter describes how to use the File Viewer, the Runtime Monitor, and the OTS Transaction Monitor to track EAServer's performance and statistics.

Using the File Viewer

The File Viewer allows you to monitor log files in the server installation, that is, the installation directory of the server to which EAServer Manager is connected.

❖ **Using the File Viewer**

1 Double-click the server for which you want to view files.

2 Highlight the Log File Viewer icon.

3 Select File | File Viewer Display.

4 The File Viewer appears. Use the tree view in the left side of the window to navigate the subdirectories of the server installation and select the log file to display.

Note In the default configuration, server log files are created in the EAServer *bin* directory.

5 Text from the selected file appears in the right side of the File Viewer. Use the following controls to configure the viewing parameters:

- Start/Stop – the Start/Stop button allows you to start and stop real-time viewing.

- Refresh – select the frequency of the refresh rate.
- Previous/Next – allows you to scroll through the current file if it is too large to view, or if you have selected a start position that does not start at the beginning of the file.
- File Size – this field displays the current file size in bytes.
- Start Position – select the start position of the file you are viewing. Your options are:
  - Tail – display as much of the end of the file as fits on the screen.
  - End – clear the file display. When you click Start, only the new entries into the file display.
  - Top – display the file starting from the top.
  - Specify Position – allows you to select a starting position by positioning a slider. Choose the incremental position from the beginning of the file.

Using the Runtime Monitor

The Runtime Monitor allows you to monitor server events and statistics, which may help you anticipate and prevent server problems.

❖ **Starting the Runtime Monitor**

1 Double-click the Servers icon.

2 Double-click the server you want to monitor.

The Runtime Monitor can connect to other EAServer instances via an IIOP listener. The server configuration identifies the host and port number to which the Runtime Monitor attempts to connect.

3 Click the Runtime Monitor icon to display these folders:

- **Packages** Monitor events and statistics for a specific package, or for all packages on the server. See “Monitoring component activity” on page 208 for more information.
- **Connection caches** Monitor a specific connection cache or statistics for all caches, as described in “Monitoring connection caches and managed connection factories” on page 209.

- **Managed Connection Factories** Monitor connection factories or statistics for all connection factories, as described in "Monitoring connection caches and managed connection factories" on page 209.
- **Network** Monitor protocol-specific session information, as described in "Monitoring network connections" on page 209.
- **Instance Pools** Monitor component instance pool activity, including the total number of instances in each pool, and the number of instances for each component assigned to the pool.
- **Active Component Models** Highlight this folder to display the component models that are in use. Information about each model is displayed in the right pane.

  If you are running Java or EJB components, you can determine the version of the server's Java virtual machine.

  If you are running PowerBuilder components using a version 9.0.1 (build 6514) or later PowerBuilder Virtual Machine (PBVM), you can determine the version and build number of the PBVM that is running the component.

Statistic counters Some monitoring folders display *counters*, which display a monitored statistic, such as component invocations or HTTP requests. When you highlight the folder, the current values of the counters display on the right side of the window. There are several types of counters:

- *Snapshot counters* represent values at a particular point in time that are likely to either increase or decrease. For example, number of sessions, number of instances active, number of active connections, and so on. Snapshot counters do not display information when set at the per-second rate; instead they display "N/A."
- *Cumulative counters* represent values that always increment and never decrement. For example, number of invocations, number of connections opened, number of network requests, bytes read and written, and so on. Cumulative counters display both per-second and counter information.
- *Peak maximum counters* represent the total value since starting the server.
- *Last maximum counters* represent the total value since starting the runtime monitor, and is not useful when there are multiple monitoring clients.

Refreshing displayed statistics You can refresh or change the counters view by choosing any of these options from the File menu:

- **Refresh** Refresh the display to obtain the latest counters.

- **Per-Second** Display the per-second values.
- **Counter** Display the accumulated value.
- **View Values** View the values in a separate dialog that automatically refreshes according to the refresh rate, which you can select from a drop-down list. If the group contains subgroups, expand those subgroups until you see the values.

Monitoring component activity

The Packages folder shows statistics about component invocations. You can view statistics for individual packages or for all packages installed in the server, including those installed indirectly as part of an installed application. Table 11-1 describes the counters.

Table 11-1: Component monitoring counters

Counter name	Description
Method invocations	Cumulative count of method invocations since the server was started.
Instances active	Snapshot count of active component instances, that is, instances currently bound to a client session.
Instances pooled	Snapshot count of pooled component instances.
Transactions completed	Cumulative count of transactions completed.
Transactions rolled back	Cumulative count of transactions rolled back.
Last maximum method invocations	Maximum count of method invocations since starting the counter view.
Last maximum instances active	Maximum count of active component instances since starting the counter view.
Last maximum instances pooled	Maximum count of pooled component instances since starting the counter view.
Last maximum transactions completed	Maximum count of transactions completed since starting the counter view.
Last maximum transactions rolled back	Maximum count of transactions rolled back since starting the counter view.
Peak maximum method invocations	Maximum method invocations since the server was started.
Peak maximum instances active	Maximum active instances since the server was started.
Peak maximum instances pooled	Maximum pooled instances since the server was started.
Peak maximum transactions completed	Maximum transactions completed since the server was started.
Peak maximum transactions rolled back	Maximum transactions rolled back since the server was started.

Monitoring connection caches and managed connection factories

The Connection Caches and Managed Connection Factory folders show statistics for cached database connections. You can display statistics for all caches or connection factories, or for each one individually. Table 11-2 describes the counters that display.

Table 11-2: Connection cache and managed connection factory statistics

Counter name	Description
Connections open	Snapshot count of open (but not necessarily active) connections.
Connections active	Snapshot count of active connections.
Connections opened	Cumulative count of the number of connections opened since the server started.
Connections closed	Cumulative count of connections closed since the server started.
Waited connection requests	Cumulative count of connection requests that required a wait because the maximum number of connections were in use. If this number is large, consider increasing the connection pool size.
No wait connection requests	Cumulative count of connection requests that required no wait.
Forced connection requests	Cumulative count of forced connection requests, that is, connections that were in excess of the configured pool size, but allocated immediately. If this number is large, consider increasing the connection pool size.
Total connection requests	Cumulative count of the total number of connection requests since the server started.
Last maximum connections active	Maximum count of active connections since starting the counter view.
Peak maximum connections active	Maximum count of active connections since the server started.

Monitoring network connections

The network connections folder allows you to monitor HTTP and IIOP connections.

HTTP and IIOP statistics

To view HTTP statistics, expand the HTTP folder, then highlight the HTTP Statistics folder to display the counters described in Table 11-3.

To view IIOP statistics, expand the IIOP folder, then highlight the IIOP Statistics folder to display the counters described in Table 11-3.

Table 11-3: HTTP and IIOP statistics counters

Counter name	Description
Sessions	Snapshot count of active sessions. This also represents the number of threads serving clients for this protocol.
Requests	Cumulative count of requests processed since the server started.
Bytes read	Cumulative count of bytes read since the server started.
Bytes written	Cumulative count of bytes written since the server started.
Last maximum sessions	Maximum number of sessions since starting the counter view.
Last maximum requests	Maximum number of requests processed since starting the counter view.
Last maximum bytes read	Maximum number of bytes read since starting the counter view.
Last maximum bytes written	Maximum number of bytes written since starting the counter view.
Peak maximum sessions	Maximum number of sessions since the server started.
Peak maximum requests	Maximum number of requests since the server started.
Peak maximum bytes read	Maximum number of bytes read since the server started.
Peak maximum bytes written	Maximum number of bytes written since the server started.
Maximum threads	The thread limit for clients of this protocol, which equals the maximum number of client sessions supported by the server configuration.

Connected users

The Connected Users folder under the IIOP folder displays the thread ID, host name, and host IP address for each IIOP client connection.

HTTP history

The HTTP History/Hits folder allows you to monitor recent request frequencies for several time intervals, as listed in Table 11-4. To enable monitoring of these statistics, expand the Runtime Monitoring/Network/HTTP History/Hits folder, then choose File | Start Data Collection. You can refresh the history view by choosing View | Refresh Folder or by pressing the F5 key.

Table 11-4: HTTP history counters

Counter name	Description
Avg hits/second last *period*	The hits-per-second rate, averaged over the specified time period. The value “Not enough data” means the specified time period has not elapsed since you enabled data collection, or that collection is not enabled.
History collection enabled?	A value of True indicates that history collection is enabled.

Using the OTS Transaction Monitor

You can monitor and administer transactions on a single server from EAServer Manager. You must be connected to the server before you can monitor or administer it.

❖ **Administering OTS/XA transactions**

1 Select OTS Transaction Viewer for a server. The dialog displays the following information for all incomplete transactions. Because the OTS Transaction Viewer captures the information about incomplete transactions at a particular moment, you need to refresh the OTS Transaction Viewer to see any changes in the incomplete transaction status or any changes that you have made directly in the OTS Transaction Viewer.

- **Transaction ID** The local identifier associated with the transaction on a specific server.
- **Family ID** The identifier of the root transaction that this transaction belongs to.

  Note Because EAServer does not support nested transactions, the Family ID is the same as the Transaction ID.

- **State** Describes the status of the transaction and can be:

State	Description
active	The transaction is currently active in the server.
inactive	The transaction is not currently active in the server.
preparing	The transaction is in the process of being prepared.
prepared	The transaction has been prepared.
committing	The transaction is in the process of being committed.
committed	The transaction has been committed.
commit_complete	The transaction has committed and all participants have been informed, but the outcome may not have been reported to the transaction originator. For example, there may have been heuristic outcomes.
before_abort	The transaction has rolled back but has not yet started the rollback process.
aborting	The transaction is in the process of rolling back.
aborted	The transaction has been rolled back.
abort_complete	The transaction has rolled back and all participants have been informed, but the outcome may not have been reported to the transaction originator. For example, there may have been heuristic outcomes.
finished	The transaction has completed.
none	The server knows about the transaction, but the server is not a participant in the transaction.
present	The transaction is active in the server but is not yet a participant in the transaction.

Note You can roll back transactions only if they are not yet in the prepared state. You can execute all other operations only on transactions that have been prepared.

- **Lock Holder** The transaction that holds the lock.
- **Lock Waiter** The transaction that is waiting for a lock to be released.
- **Level** An integer that represents the level of a nested transaction. Since EAServer does not support nested transactions, this field is always 1.

2 In the dialog that displays, select a transaction and click one of these buttons:

- **Refresh List** Refresh the list of all unfinished transactions.

- **Rollback** Roll back the entire transaction, including any sub-transactions.

  Note EAServer does not currently support nested OTS/XA transactions (also called subtransactions).

- **Force Complete** Force the transaction's rollback or commit processing (whichever outcome the transaction coordinator determined for the transaction) to be completed.
- **Heuristic Commit** Perform a heuristic commit.
- **Heuristic Rollback** Perform a heuristic rollback.
- **Global Info** Displays the global transaction identifier and the identifier of the application that started the transaction.

CHAPTER 12

Using jagtool and jagant

jagtool is a command line interface that allows you to automate some of EAServer's development and deployment tasks. You can use jagtool from the command line, from scripts or makefiles, or with Jakarta Ant.

This chapter contains instructions on how to use jagtool, either by itself, or with jagant.

Working with jagtool

Before using jagtool, make sure that:

- The JAGUAR environment variable is set.
- **UNIX** *$JAGUAR/bin* is added to your path.
- **Windows** *%JAGUAR%\bin* is added to your path.

Use the following scripts to run jagtool:

- **UNIX** *$JAGUAR/bin/jagtool*
- **Windows** *%JAGUAR%\bin\jagtool.bat*

jagtool syntax

The syntax for jagtool is:

jagtool [*connect-args* | *local-args*] [*log-arg*] [*command*]

Where:

- *connect-args* is a list of arguments required to run in connected mode.
- *local-args* is a list of arguments required to run in local mode.
- *log-arg* is an optional argument to specify a file name to record jagtool output. If you do not specify a file name, output goes to the standard output device. Specify a file name with the `-logfile` ***`filename`*** argument or `-l` ***`filename`***.
- *command* is a jagtool command described in "jagtool commands" on page 229.

Local versus connected mode

You can run jagtool in two modes. In connected.mode, jagtool connects to a server which can be running locally or on a remote machine. In local mode, jagtool does not require a connection to a server, but must be run on a machine with file system access to the EAServer installation.

Using connected mode

Connected mode is jagtool's default mode of operation. All commands can run in connected mode. When using connected mode, specify the arguments listed in Table 12-1.

Table 12-1: Jagtool connection arguments

Parameter	To specify
`-h` *`hostname`* or `-host` *`hostname`*	Server host name. If not specified, the default is the value of the JAGUAR_HOST_NAME environment variable.
`-n` *`port`* or `-port` *`port`*	Server IIOP port number. If not specified, the default is 9000.
`-u` *`name`* or `-user` *`name`*	User name. If not specified, the default is "jagadmin".
`-p` *`password`* or `-password` *`password`*	Password. If not specified, the default is "" (no password).

For example, to connect to the server running on *eclipse* at port *9005*, using account *jagadmin* with password *secret* enter:

```
jagtool -h eclipse -n 9005 -p secret
```

You can omit the *-u* flag because *jagadmin* is the default user name.

Unless otherwise specified in the command reference page, all commands can run in connected mode.

Using local mode

Local mode allows you to configure an EAServer installation without requiring a connection to a server. Local mode is helpful in situations where it is not convenient to start a server, for example, when using jagtool to configure new installations. Not all commands can run in local mode. To see whether a command supports local mode, check the syntax listing for the command in this chapter or check the help output for the command.

To run in local mode, specify the arguments in Table 12-2.

Table 12-2: Jagtool local-mode arguments

Parameter	To specify
`-local`	Specifies whether to run in local mode, without a connection to the server. If not specified, jagtool requires a connection to a running server.
`-server` *servername*	Specifies the name of the server to use when running in local mode. Specify the name of the server that you would connect to if running in connected mode. If you do not specify a server name, the default is Jaguar.

Local mode also requires the following:

- You must be logged in to the operating system as a user with read and write privileges on the EAServer installation directory and all subdirectories.
- You must set the JAGUAR environment variable set to specify the local installation directory.

When running jagtool or jagant locally, the user name and password arguments are optional. If supplied, these arguments are ignored.

Not all commands can run in local mode. The reference page for each command indicates whether the command supports local mode.

Entity identifiers

Many jagtool commands take one or more entity identifiers as arguments. An entity identifier is a string of the form *EntityType:EntityName* that uniquely identifies an entity in the repository.

Table 12-3 provides examples of entity identifiers for each entity type.

Table 12-3: Example entity identifiers

Entity Identifier	Specifies
`Agent:agent1`	Agent named agent1.
`Application:estore`	Application named estore.
`ApplicationClient:estore/PetStoreClient`	Application client PetstoreClient in application estore.
`Cluster:TheBigCluster`	The cluster named TheBigCluster.
`Component:SVU/SVULogin`	Component named SVULogin that is installed in the SVU package. The package name is included because EAServer components always reside in packages.
`ConnCache:JavaCache`	Connection cache named JavaCache.
`Connector:BlackBoxLocalTx`	J2EE connector named BlackBoxLocalTx.
`DatabaseType:Sybase_ASA`	Database type definition named Sybase_ASA.
`EntityCollection:MyEntityCollection`	The entity collection named “MyEntityCollection.”
`Filter:WebTier/MyFilter`	The servlet filter MyFilter installed in the Web application WebTier.
`InstancePool:MyPool`	The named instance pool MyPool.
`Listener:Jaguar/iiops1`	The network listener iiops1 installed in the server named Jaguar. When specifying a listener name, use the server name and the listener name as displayed in EAServer Manager.
`ManagedConnectionFactory:BlackBoxLocalTx/EASDemo`	The managed connection factory EASDemo in the J2EE connector named BlackBoxLocalTx.
`Method:SVU/SVULogin/isLogin`	The isLogin method of component SVULogin in package SVU.
`Package:SVU`	The package named SVU.
`Role:MyRole`	The role named MyRole.
`Security:sample1`	The security entity named “sample1.”

Entity Identifier	Specifies
`Server:Jaguar`	The server named Jaguar.
`Service:MyPack/MyComp`	The service component MyComp, installed in package MyPack. Use this syntax to install or remove service components from a server.
`Servlet:StandAloneServlet`	The servlet named StandaloneServlet. This syntax is valid only for servlets that are not installed in a Web application.
`Servlet:MyWebApp/MyServlet`	The servlet named MyServlet in Web application named MyWebApp. You must use this syntax for servlets that are installed in a Web application.
`WebApplication:WebTier`	The Web application named Web tier.

Not all jagtool commands support every type of entity in the repository. For example, the refresh command is not supported for the Listener entity type.

When a command specifies an invalid entity type, an appropriate error message is displayed.

jagtool and jagant

jagant lets you run jagtool commands from Ant build files. This powerful feature allows you to write build files that automate many development and deployment tasks.

Jakarta Ant is a Java-based build tool developed by the Apache Jakarta project. To obtain Ant software and documentation, see the Ant Web site at http://jakarta.apache.org/ant/. Ant functions similarly to other build tools (such as *make*, *gnumake*, or *jam*) but is platform-independent, extending Java classes rather than OS-specific shell commands. Ant includes a number of tasks that are frequently used to perform builds, including compiling Java files and creating JAR files. It also includes common functions such as copy, delete, chmod, and so on.

Ant build files (similar to a *make* file) are written in XML. Like *make*, Ant build files can include targets that perform a series of tasks. Instead of extending shell commands, Ant's build file calls out a target tree where various tasks are executed. Each task is run by an object that implements a particular task interface.

Setting up your environment

Install Ant and read the accompanying documentation.

The jagant script requires a full JDK installation. If you are running jagant from an EAServer client install, make sure you have installed the full JDK. By default, only the JRE files are installed.

Before running jagant, verify that:

- The JAGUAR environment variable is set.
- A full JDK installation is present.
- Jakarta Ant is installed on your system.

  By default, jagant searches for Jakarta Ant in *%JAGUAR%\jakarta-ant* (Windows) or *$JAGUAR/jakarta-ant* (Solaris). If you install Jakarta Ant in a different location, set the ANT_HOME environment variable before running the jagant script to reflect the change.

  You can also set ANT_HOME in the user environment file, *%JAGUAR%\bin\user_setenv.bat* (Windows) or *$JAGUAR/bin/user_setenv.sh* (UNIX). The jagant script checks the user environment file each time it runs.
- If you are using jagant to compile JSP files with the compilejsp command, modify the CLASSPATH setting for the Ant scripts, adding the location of the *xalan.jar* and *crimson.jar* files that are included with EAServer. For example, if using Windows, edit the *ant.bat* file, and change the code under the `:runAnt` label to read:

```
:runAnt
set JAGUAR=EAServer install directory
set LOCALCLASSPATH=%JAGUAR%\java\classes\crimson.jar;%LOCALCLASSPATH%
set LOCALCLASSPATH=%JAGUAR%\java\classes\xalan.jar;%LOCALCLASSPATH%
%_JAVACMD% -classpath %LOCALCLASSPATH% -Dant.home="%ANT_HOME%"
%ANT_OPTS% org.apache.tools.ant.Main %ANT_CMD_LINE_ARGS%
goto end
```

Or on UNIX, change the last line to read like these lines:

```
JAGUAR=EAServer install directory
LOCALCLASSPATH=$JAGUAR/java/classes/crimson.jar:$LOCALCLASSPATH
LOCALCLASSPATH=$JAGUAR/java/classes/xalan.jar:$LOCALCLASSPATH
$JAVACMD -classpath "$LOCALCLASSPATH" -Dant.home="${ANT_HOME}" $ANT_OPTS
org.apache.tools.ant.Main "$@"
```

jagant scripts

The following scripts are provided for running Ant with jagtool commands:

- **Windows** *%JAGUAR%\bin\jagant.bat*
- **UNIX** *$JAGUAR/bin/jagant.sh*

jagant syntax

The jagant script uses this syntax:

jagant *[ant_options]*

where *ant_options* are any options and commands supported by Ant; see the Ant documentation for details on these options.

You may frequently use the *-buildfile* flag. Using *-buildfile* lets you specify a location other than the default for the Ant XML build file.

The Ant build file

The EAServer installation includes a sample XML build file. You can find it in *%JAGUAR%\sample\jagtool\sample.xml* (UNIX) or *$JAGUAR/sample/jagtool/sample.xml* (Windows).

This file contains targets for:

- Refreshing a package
- Restarting a server
- Exporting a package to a JAR file

- Deploying a J2EE EAR file
- Creating a package
- Setting package properties

For example, to run jagant with the sample build file to refresh a package named SVU, enter this command all on one line:

```
jagant -buildfile %JAGUAR%\sample\jagtool\sample.xml refresh_svu
```

In this example, jagant is invoked with the specified build file. The target, *refresh_svu*, is defined in the build file as:

```
<!--refresh package svu -->
<target name="refresh_svu" depends="connect">
  <jag_refresh entity="Package:SVU" />
</target>
```

The refresh_svu target invokes the jag_refresh command to refresh the package named SVU. The refresh_svu target depends on the connect target. This dependency causes the connect target to run before the refresh_svu target. The connect target is defined as follows:

```
<!-- connect -->
<target name="connect">
<jag_connect host="$(jaguar.host)"port="$(jaguar.port)" user="$(jaguar.user)"
password="$(jaguar.password)" />
</target>
```

The connect target invokes the jag_connect command to open a connection with the server. The host name, port number, user name and password are variables. They are defined earlier in the build file as follows:

```
<!-- global properties for this build -->
<property name="jaguar.host" value="yourMachine" />
<property name="jaguar.port" value="9000" />
<property name="jaguar.user" value="jagadmin" />
<property name="jaguar.password" value="" />
```

You can override these values at the command line using the Ant -D option. This is typically done to specify the password, so that it is not stored directly in the build file. For example (entered all on one line):

```
jagant -Djaguar.host=eclipse -Djaguar.port=9005 -Djaguar.password=jagpass
-buildfile %JAGUAR%\sample\jagtool\sample.xml refresh_svu
```

This command connects to the server with a host name of *eclipse* on port *9005*, with the user name *jagadmin* and the password *jagpass*. The default user of *jagadmin* is still used because it was not overridden at the command line.

Registering jagtool commands in the Ant build file

Each build file that invokes jagtool commands must include definitions for those commands. This is done by including an Ant *taskdef* directive for each jagtool command. You can see these directives in the sample build file. For example:

```
<!-- task definitions -->
<taskdef name="jag_connect"
classname="com.sybase.jaguar.management.jagtool.ant
.ConnectTask" />
<taskdef name="jag_copy"
class="com.sybase.jaguar.management.jagtool.ant.Cop
yTask" />
```

Each definition includes the name of the command and the location of the class file it invokes.

Using the *jag_connect* command

In build files, use the jag_connect command to connect to a server or to specify the server name for local mode. You cannot use jag_connect from the command line; instead use the connection or local-mode arguments described in “Local versus connected mode” on page 216.

jag_connect must be executed before any other jagtool commands in the build file. jag_connect can be included directly in any target, or in a “connect” target that other targets list as a dependency.

As with jagtool, you can run jagant in local or connected mode. “Local versus connected mode” on page 216 explains the difference.

Using jagant in connected mode

To run jagant in connected mode, specify these options for the jag_connect command:

- **host** The name of the host where EAServer is running.

- **port** The port number for the server. The default is *9000*.
- **user** The user name used to connect. The default is *jagadmin*.
- **password** The password used to connect. The default is no password.
- **logfile** The log file for the connection attempt. The default is *System.out*.

For example, this sample project defines a connect task to connect to the machine “myhost” at port 9000, logging in as “jagadmin,” and runs the jag_list command over the connection:

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

<!DOCTYPE project [
<!ENTITY jagtasks SYSTEM "file:./jagtasks.xml">
]>

<project name="local_sample" default="list_packages" basedir=".">

    <!-- include Jaguar task definitions -->
    &jagtasks;

    <!-- connect -->
    <target name="connect">
        <jag_connect host="myhost" port="9000" user="jagadmin" password="" />
    </target>

    <!-- list packages in the server -->
    <target name="list_packages" depends="connect">
        <jag_list type="Package" />
    </target>

    <!-- list the properties of package CtsSecurity -->
    <target name="CtsSecurity_props" depends="connect">
        <jag_props entity="Package:CtsSecurity" />
    </target>

</project>
```

Using jagant in local mode

To define a jag_connect task to run in local mode, set the localServer option to the name of the server to use when running in local mode. Specify the name of the server that you would connect to if running in connected mode. If you specify this option, the connected-mode arguments are ignored and jagant runs in local mode. For example, this sample project defines a connect task to run in local mode, then runs the jag_list command in the scope of the local-mode connection:

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

<!DOCTYPE project [
<!ENTITY jagtasks SYSTEM "file:./jagtasks.xml">
]>

<project name="local_sample" default="list_packages" basedir=".">

    <!-- include Jaguar task definitions -->
    &jagtasks;

    <!-- connect -->
    <target name="connect">
        <jag_connect localServer="Jaguar" />
    </target>

    <!-- list packages in the server -->
    <target name="list_packages" depends="connect">
        <jag_list type="Package" />
    </target>

</project>
```

Using multiple connections

You can use multiple jag_connect commands in a single build file, which allows you to execute jagtool commands for different servers. Each time jag_connect is executed, the current connection or local-mode session is closed and a new one is opened. For example, this code restarts two servers:

```
<target name="restart_all_servers">
<jag_connect host="host1" password-="jagpass" />
<jag_restart />
<jag_connect host="host2" password="jagpass" />
<jag_restart /></target>
```

A connection to *host1* is opened and the server on *host1* is restarted. Then the connection is closed, a connection is opened to *host2*, and the server on *host2* is restarted. The port number for both servers is *9000* and the user name is *jagadmin* (the defaults). The password for both servers is *jagpass*, and the log file is *System.out*.

XML configuration files

Rather than using jagtool or jagant to configure an entity's properties individually, you can define entity properties in XML. You can define and configure multiple entities in one XML file.

With jagtool, you can use the exportconfig command to create an XML configuration file that replicates an existing entity, and the configure command to apply the settings in an XML file to the repository.

You can also embed EAServer XML configuration files in J2EE-format archive files for components, applications, Web applications, and connectors, as described in "Using EAServer configuration files in J2EE archives" on page 196.

Format of the XML configuration file

Configuration files must use this DTD, located in your EAServer installation:

```
java/lib/dtds/sybase-easerver-config_1_0.dtd
```

sybase-easerver-config element

The sybase-easerver-config element is the root element and can contain zero or more macro elements followed by one or more configure elements. You can use the description attribute for comment text; this text does not affect any properties set in the repository. Here is an example:

```
<sybase-easerver-config description="Configuring EAServer properties">
... deleted ...
</sybase-easerver-config>
```

macro elements

You can use macro elements to define abbreviations for commonly used strings in the XML file, such as the com.sybase.jaguar.component prefix used in most component property names. macro elements have optional begin and end attributes to specify the delimiters. If you do not specify them, the default begin and end delimiters are ${ and }, respectively. The macro element can contain one or more definition elements to specify the macro abbreviations, for example:

```
<macro begin="${" end="}">
    <definition name="comp" value="com.sybase.jaguar.component"/>
    <definition name="desc" value="com.sybase.jaguar.description"/>
</macro>
```

configure elements

A configure element creates and updates an entity. The value of the type attribute specifies the operation to perform, as follows:

Value of type attribute	Action
create	Create a new entity with the specified name and with the specified properties. The operation fails if the entity already exists.
update	Update the specified properties for an existing entity. The operation fails if the entity does not exist.
delete	Deletes the specified property settings from an existing entity. The operation fails if the entity does not exist.

If you are embedding an XML file in a J2EE archive to configure entities defined in the archive, use the update operation. Entities defined in the file will exist when the XML configuration file is applied.

The entity attribute specifies the entity to operate on, using the format described in “Entity identifiers” on page 218. The configure element can contain 0 or more property elements to configure the entity’s properties.

Here is an example configure element:

```
<configure type="create" entity="Package:DocTest">
  <property name="${desc}" value="New package" />
</configure>
```

property elements

The property element specifies a property setting for the entity, for example:

```
<property name="com.sybase.jaguar.component.stateless" value="true" />
```

Special characters

For special characters in property values, use the appropriate XML entity identifier, such as `&quot;` for the double-quote symbol (").

Sample configuration file

Here is a sample configuration file that defines a package, *DocTest*, and a component in the package, *FooComponent*:

```
<!DOCTYPE sybase-easerver-config PUBLIC '-
//Sybase, Inc.//DTD EAServer configuration 1.0//EN' 'http://www.sybase.com/dt
ds/easerver/sybase-easerver-config_1_0.dtd'>

<sybase-easerver-config description="Configuring EAServer properties">
  <macro begin="${" end="}">
        <definition name="comp" value="com.sybase.jaguar.component"/>
        <definition name="desc" value="com.sybase.jaguar.description"/>
  </macro>
  <configure type="create" entity="Package:DocTest">
    <property name="${desc}" value="New package" />
  </configure>
  <configure type="create" entity="Component:DocTest/FooComponent">
    <property name="${comp}.debug" value="true" />
    <property name="${comp}.name" value="DocTest/FooComponent" />
    <property name="${desc}" value="New description" />
    <property name="com.sybase.jaguar.component.type" value="java" />
    <property name="com.sybase.jaguar.component.control" value="JaguarEJB::Se
rverBean" />
    <property name="com.sybase.jaguar.component.sharing" value="true" />
    <property name="com.sybase.jaguar.component.roles" value="" />
    <property name="com.sybase.jaguar.component.tx_outcome" value="always" />
    <property name="com.sybase.jaguar.component.pooling" value="false" />
    <property name="com.sybase.jaguar.component.java.class" value="com.sybase
.jaguar.sample.events.StockManagerImpl" />
    <property name="com.sybase.jaguar.component.thread.safe" value="true" />
    <property name="com.sybase.jaguar.component.stateless" value="false" />
    <property name="com.sybase.jaguar.component.java.classes" value="" />
```

```
    <property name="com.sybase.jaguar.component.interfaces" value="EventSampl
es::StockManager" />
    <property name="com.sybase.jaguar.component.ids" value="IDL:EventSamples/
StockManager:1.0" />
    <property name="com.sybase.jaguar.component.transient" value="false" />
    <property name="com.sybase.jaguar.component.auto.failover" value="false"
/>
    <property name="com.sybase.jaguar.component.tx_type" value="not_supported
" />
    <property name="com.sybase.jaguar.component.tx_control" value="true" />
    <property name="com.sybase.jaguar.component.refresh" value="true" />
    <property name="com.sybase.jaguar.component.model" value="com" />
    <property name="com.sybase.jaguar.component.tx_vote" value="false" />
  </configure>
 </sybase-easerver-config>
```

jagtool commands

This section contains information on jagtool commands, and lists the commands that jagtool accepts. Each command has a brief description, a list of options, and an example of its usage.

Each command has its own section heading (the text in the far left margin). Each command section contains a description of the command, its syntax, a list of options, and an example of its use at the command line and in Ant build files.

compilejsp

Description

Compiles JSP files.

Syntax

Local mode support: Yes.

Connected mode support: No.

Command line:

compilejsp
local-args
[-webapp *WebAppName*]
[-uriroot *directory*]

[-outputdir *directory*]
[-verbose *true*|*false*]
[-debug *true*|*false*]
[-keep *true*|*false*]
[-force *true*|*false*]
[-servername *server*]
[*file1 file2 ...*]

Ant build file:

<jag_compilejsp
[webapp="*WebAppName*"]
[uriroot="*directory*"]
[outputdir="*directory*"]
[verbose="*true*|*false*"]
[debug="*true*|*false*"]
[keep="*true*|*false*"]
[force="*true*|*false*"]
[servername="*server*"]
[jspfile="*file*"] />

Option	Description	Required
local-args	Arguments to run in local mode. See "Using local mode" on page 217.	Yes
`webapp`	Compiles the JSP files in the specified Web application. Uses the JAGUAR environment variable to locate the Web application directory. Compiles all JSP files in the Web application unless specific files are specified. You must specify one of the `-webapp` or `-uriroot` options, but not both. You must specify the `-webapp` option to compile JSPs that require tag library mappings defined in the Web application properties file.	No
`uriroot`	Compiles the specified JSP files in the given directory, which must be specified as a complete path. If no files are specified, compiles all JSP files in the directory and its subdirectories.	No
`outputdir`	The full path of the output directory for generated Java source and compiled classes. If the `-webapp` option is specified, the default is this subdirectory of your EAServer installation directory: `work\`*`server`*`\Servlet\WebApp-`*`WAName`* Where *server* is the name of your server, and *WAName* is the name of your Web application. To run the JSPs in EAServer, class files must be in this directory. If the `-webapp` option is not specified, the default is the current directory.	No
`verbose`	Execute in verbose mode. The default is `false`.	No

Option	Description	Required
`debug`	Compile the generated Java files with debugging information. The default is `false`.	No
`keep`	Keep the generated Java source files rather than deleting them. The default is `true`.	No
`force`	Force compilation of JSP files, regardless of whether the class file is newer than the JSP source file. The default is `true`.	No
`servername`	The server for which the JSP should be compiled. The default is Jaguar.	No
file1 file2 ...	When using the command line, the list of files to compile, with paths specified relative to the Web application root directory, if using the webapp option, or the specified directory, if using the uriroot option. If no files are specified, all files in the specified Web application or directory are compiled.	No
`jspfile`	When using Ant, the file to compile with path relative to the relative to the Web application root directory, if using the webapp option, or the specified directory, if using the uriroot option. If no file is specified, all files in the specified Web application or directory are compiled.	No

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This command line example compiles the files *jsp/file1.jsp* and *jsp/file2.jsp* in the Web application MyWebApp:

```
jagtool compilejsp -webapp MyWebApp jsp/file1.jsp jsp/file2.jsp
```

The output goes to the EAServer directory:

```
work/server/Servlet/WebApp-MyWebApp
```

Example 2 This command line example recursively compiles all JSPs in the Web application MyWebApp:

```
jagtool compilejsp -webapp MyWebApp
```

The output goes to the EAServer directory:

```
$JAGUAR/work/server/Servlet/WebApp-MyWebApp
```

Where *server* is the name of the server you are connected to.

Example 3 This command line example compiles *file1.jsp* in the Web application MyWebApp:

```
jagtool compilejsp -webapp MyWebApp -outputdir c:\temp file1.jsp
```

The output goes to *c:\temp*.

Example 4 This command line example compiles *file1.jsp* in the directory *c:\webapps\MyWebAppDir*:

```
jagtool compilejsp -uriroot c:\webapps\MyWebAppDir file1.jsp
```

The output goes to the current directory.

Example 5 This command line example recursively compiles all JSPs in the directory *c:\webapps\MyWebAppDir*:

```
jagtool compilejsp -uriroot c:\webapps\MyWebAppDir -outputdir c:\temp
```

The output goes to *c:\temp*.

Example 6 This Ant build file example defines a target to compile two JSP files:

```
<target name="compilejsp_test" >
  <jag_compilejsp Jspfile="file1.jsp" verbose="false"
Uriroot="D:\EAS\Sample\jagtool" />
  <jag_compilejsp Jspfile="file2.jsp" verbose="false"
Uriroot="D:\EAS\Sample\jagtool" />
</target>
```

Usage

You must run compilejsp in local mode. If running jagtool, see "Using local mode" on page 217. If running jagant, see "Using jagant in local mode" on page 225.

To use the compilejsp command in Ant, you must add the location of the *xalan.jar* and *crimson.jar* files that are included with EAServer to the Ant CLASSPATH. See "Setting up your environment" on page 220 for more information.

You can also compile JSPs with the EAServer jspc script. The jspc script invokes jagtool to compile JSPs.

See also

Chapter 24, "Creating JavaServer Pages," in the *EAServer Programmer's Guide*

configure

Description

Configures or defines entities in the repository by reading properties from an XML file.

Syntax

Local mode support: Yes.

Command line:

configure
[*connect-args* | *local-args*]
[-verbose *true*|*false*]
filename

Ant build file:

<jag_configure [verbose="*true*|*false*"] file="*filename*" />

Where:

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
`verbose`	Execute in verbose mode. The default is `false`.	No
filename	XML file to read commands from.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Usage

The configure command allows you to define and configure entities with an XML file.

See also

exportconfig, "XML configuration files" on page 226

copy

Description

Copies an entity in the repository.

Syntax

Local mode support: Yes.

Command line:

copy [*connect-args* | *local-args*] *source target*

Ant build file:

<jag_copy source="*source*" target="*target*" />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
source	Entity identifier for the entity being copied.	Yes
target	Entity identifier for the new copy.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

This example creates a copy of the entity *Package:SVU* and places it in the entity *Package:SVU_2*.

- Command line:

```
jagtool copy Package:SVU Package:SVU_2
```

- Ant build file:

```
<jag_copy source="Package:SVU" target="Package:SVU_2" />
```

create

Description

Creates a new entity in the repository.

Syntax

Local mode support: Yes.

Command line:

create [*connect-args* | *local-args*] *entity* [*file*]

Ant build file, specifying properties from an optional file:

<jag_create entity="*entity*" [file="*file*"] />

Ant build file, specifying properties directly:

```
<jag_create entity="entity" >
  <property name="name" value="value" />
    ....
</jag_create>
```

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or run in local mode. See "Local versus connected mode" on page 216.	Yes
entity	The name of the entity being created, in the form *EntityType:EntityName*.	Yes
file	An optional file containing properties for the entity. The file must specify properties in the form of an EAServer repository properties file.	No
name	The property name. In an Ant build file, you may specify multiple properties as `<property>` elements.	When setting properties directly in Ant.
value	The property value.	When setting properties directly in Ant.

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example creates a package named *NewPackage* with the properties defined in the file *NewPackage.props*.

- Command line:

```
jagtool create Package:NewPackage NewPackage.props
```

- Ant build file:

```
<jag_create entity="Package:NewPackage" file="NewPackage.props" />
```

Example 2 This example creates a package named NewPackage2 and sets the given properties. This alternate syntax allows properties for the new entity to be specified in the command.

- Ant build file:

```
<jag_create entity="NewPackage2">
 <property name="com.sybase.jaguar.description" value=Sample Package" />
```

```
 <property name="com.sybase.jaguar.package.roles" value="role1" />
</jag_create>
```

Example 3 This example creates a listener called *MyListener* for the Jaguar server:

- Command line:

```
jagtool create Listener:Jaguar/MyListener
```

- Ant build file:

```
<jag_create entity="Listener:Jaguar/MyListener" />
```

Usage

The create command does not perform the installation steps required to run an entity in a particular server. For example, a package or application must be installed in a server before running on that server, and a listener must be installed in the server that it is associated with. Use the install command to install entities into a parent entity.

See also

delete, install, jmscreate

delete

Description

Deletes an entity from the repository.

Syntax

Local mode support: Yes.

Command line:

delete [*connect-args* | *local-args*] [-type type] *entity*

Ant build file:

<jag_delete [type="*type*"] entity="*entity*" />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
type	Specify `simple` for simple deletion (properties files only) or `full` to delete all files that were created when the entity was deployed. The default is `simple`. For more information, see "Deleting packages, Web applications, and applications" on page 199.	
entity	The identifier for the entity being deleted.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example deletes Package:SVU from the repository.

- Command line:

```
jagtool delete Package:SVU
```

- Ant build file:

```
<jag_delete entity="Package:SVU" />
```

Note When a package is deleted, it is also removed from any servers on which it is installed.

Example 2 This example deletes the SVULogin component from Package:SVU.

- Command line:

```
jagtool delete Component:SVU/SVULogin
```

- Ant build file:

```
<jag_delete entity="Component:SVU/SVULogin" />
```

deploy

Description

Deploys a J2EE EAR file, J2EE WAR file, J2EE JAR file, or Jaguar JAR file to the server.

Syntax

Local mode support: Yes.

Command line:

deploy
[*connect-args* | *local-args*]
[-type *filetype*]
[-stubsandskels *true*|*false*]
[-jagjartype *jartype*]
[-install *true*|*false*]

[-strategy *strategy*]
[-verbose *true*|*false*]
[-interoperablenaming *true*|*false*]
[-setjarfilepackagenaming=”*true*|*false*”]
filename

Ant build file:

<jag_deploy
[type="*filetype*"]
[stubsandskels="*true*|*false*"]
[jagjartype="*jartype*"]
[install="*true*|*false*"]
[strategy="*strategy*"]
[verbose="*true*|*false*"]
[interoperablenaming=”*true*|*false*”]
[setjarfilepackagenaming=”*true*|*false*”]
file="*filename*” />

Option	Description	Default	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	-	Yes
`type`	The type of file deployed: • ear • war • ejbjar • jagjar • rar	ear	No
`stubsandskels`	Indicates whether stubs and skeletons should be generated for EJB files. This flag is applicable only to EAR and EJB files.	true	No
`jagjartype`	When *type* is `jagjar`, specifies the contents of the Jaguar JAR file. Options are: • Application • Connector • EntityCollection • Package • WebApplication	Application for type `jagjar`; otherwise, none	Only for type `jagjar`
`install`	Indicates whether installed entities should be automatically deployed in servers.	true	No

Option	Description	Default	Required
`strategy`	Indicates the deployment strategy. Options are: • **full** IDL is generated for everything in the deployment tree. • **incremental** IDL is generated for everything that has changed. • **optimistic** Used for implementation changes (for example, if you have changed the contents of a method without changing its prototype). You can specify *incremental* when the methods, fields, interfaces, or superclass of a class have changed. Specify *full* when other details have changed or IDL has been deleted.	incremental	No
`interoperablenaming`	Enables interoperable naming. This option is available only when you are running JDK 1.3—see "Interoperable naming" on page 111.	true	No
`setjarfilepackagenaming`	For EAR or EJB-JAR files, specifies how newly created packages are named, as follows: • `true` indicates that entities use the name of the archive file that they are imported from. • `false` indicates that entities use the `display-name` element in the deployment descriptor.	false	No
`verbose`	Indicates that output during deployment is verbose.	false	No
filename	The name of the deployed file.	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This command runs in local mode and deploys the EJB-JAR file named *myejb.jar* into the server named FooServer:

```
jagtool -local -server FooServer deploy -type ejbjar myejb.jar
```

Example 2 This example deploys the J2EE EAR file *eastore.ear* to the server:

- Command line:

```
jagtool deploy -type ear e:\temp\estore.ear
```

- Ant build file:

```
<jag_deploy type="ear" file="e:\temp\estore.ear" />
```

Example 3 This example deploys the JAR file *AuthServiceDemo.jar* to the server:

- Command line:

```
jagtool deploy -type jagjar -jagjartype Package /tmp/AuthServiceDemo.jar
```

- Ant build file:

```
<jag_deploy type="jagjar" jagjartype="Package"
file="/tmp/AuthServiceDemo.jar" />
```

See also

export, gen_stubsandskels, install,
Chapter 9, “Importing and Exporting Application Components”

ejbref

Description

Sets the value of an EJB reference.

Syntax

Local mode support: Yes.

Command line:

ejbref
[*connect-args* | *local-args*]
entity
[-localref *true*|*false*]
-refname *name*
-value *value*

Ant build file:

<jag_ejbref [localref=”*true*|*false*”] entity=”*entity*” refname=”*name*” value=”*value*” />

Option	Description	Default	Required
connect-args| *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	-	Yes
entity	The entity identifier in the form *EntityType*:*EntityName*.	-	Yes

Option	Description	Default	Required
`localref`	Whether setting a local-interface reference or a remote-interface reference. `true` indicates a local-interface reference.	`false`	No
`refname`	The name of an EJB reference for the given entity.	-	Yes
`value`	The value for the EJB reference.	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

This example sets the value of an EJB reference in the component TheCustomer in the package Customer_Component. The EJB reference ejb/account/Account is set to the value of "Customer_Component/TheAccount".

- Command line (all on one line):

```
jagtool ejbref Component:Customer_Component/TheCustomer
-refname ejb/account/Account
-value Customer_Component/TheAccount
```

- Ant build file:

```
<jag_ejbref entity="Component:Customer_Component/TheCustomer"
refname="ejb/account/Account" value="Customer_Component/TheAccount" />
```

enventry

Description

Sets the value of a J2EE environment entry.

Syntax

Local mode support: Yes.

Command line:

enventry
[*connect-args* | *local-args*]
entity
-entryname *name*
-value *value*

Ant build file:

<jag_enventry entity="*entity*" entryname="*name*" value="*value*" />

Option	Description	Default	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	-	Yes
entity	The entity identifier in the form *EntityType*:*EntityName*.	-	Yes
`entryname`	The name of an environment entry for the given entity.	-	Yes
`value`	The value for the environment entry.	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

This example sets the value of an environment entry in the component TheOrder in the package Customer_Component. The value of environment entry ejb/order/OrderDAOClass is set to "com.sun.j2ee.blueprints.customer.order.dao.OrderDAOSybase".

- Command line (all on one line):

```
jagtool enventry Component:Customer_Component/TheOrder -entryname
ejb/order/OrderDAOClass -value
com.sun.j2ee.blueprints.customer.order.dao.OrderDAOSybase
```

- Ant build file:

```
<jag_enventry entity="Component:Customer_Component/TheOrder"
entryname="ejb/order/OrderDAOClass"
value="com.sun.j2ee.blueprints.customer.order.dao.OrderDAOSybase" />
```

exists

Description

Determines whether or not a specified entity is present in the configuration repository.

Syntax

Local mode support: Yes.

Command line:

exists [*connect-args* | *local-args*] *entity*

Ant build file:

<jag_exists entity="*entity*" property="*ant_property*">

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
entity	The entity identifier in the form *EntityType*:*EntityName*.	Always
ant_property	The name of the Ant build property to set if the entity exists.	When using Ant

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example determines whether the component SVULogin in package SVU exists:

- Command line:

```
jagtool exists Component:SVU/SVULogin
```

- Ant build file:

  This example does the same, and sets the property `svulogin.exists` if the component exists. If it does not exist, the property is not set.

```
<jag_exists entity="Component:SVU/SVULogin" property="svulogin.exists"
/>
```

export

Description

Exports an entity to a J2EE format EAR, WAR, J2EE JAR, or RAR file, or to an EAServer format JAR file.

Syntax

Local mode support: Yes.

Command line:

export [*connect-args* | *local-args*] [-dir *dirname*] [-jagjar *true*|*false*] \
[-xmlconfig=*true*|*false*] [-emptycachetags=*true*|*false*] *entity*

Ant build file:

<jag_export [dir="*dirname*"] [jagjar="*true*|*false*"] [-xmlconfig="*true*|*false*"] -[emptycachetags="*true*|*false*"] entity="*entity*" />

Option	Description	Default	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	-	Yes
`dir`	The directory where the file is created.	Current directory	No
`jagjar`	Export to a Jaguar JAR file rather than a J2EE archive file.	true	No
`xmlconfig`	When exporting a J2EE archive, whether to include the EAServer-specific *sybase-easerver-config.xml* file. See "Using EAServer configuration files in J2EE archives" on page 196 for more information.	true	No
`emptycachetags`	When exporting a J2EE archive, whether to export a no-operation version of the EAServer partial page caching tag library with Web applications to allow portability to other J2EE application servers that do not support this tag library	false	No
entity	Entity identifier for the entity being exported.	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.

Return value	Indicates
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example exports the application named "estore" to a J2EE EAR file in the *e:\temp* directory:

- Command line:

```
jagtool export -dir e:\temp Application:estore
```

- Ant build file:

```
<jag_export dirname="e:\temp" entity="Application:estore" />
```

Example 2 This example exports the SVU package to a JAR file in the current directory:

- Command line:

```
jagtool export -jagjar true Package:SVU
```

- Ant build file:

```
<jag_export jagjar="true" entity="Package:SVU" />
```

See also

deploy,
Chapter 9, "Importing and Exporting Application Components"

exportconfig

Description

Creates an XML configuration file that matches the configuration of an existing entity.

Syntax

Local mode support: Yes.

Command line:

exportconfig [*connect-args* | *local-args*] [-dir *dirname*] \
[-configtype *update*|*create*] [-verbose *true*|*false*] \
[-easerverpropsonly *true*|*false*] *entity*

Ant build file:

<jag_exportconfig [dir="*dirname*"] [configtype="*update*|*create*"] [verbose=*true*|*false*"] [easerverpropsonly "*true*|*false* "] entity="*entity*" />

Option	Description	Default	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	-	Yes
`dir`	The directory where the file is to be created.	The current directory.	No
`configtype`	Specifies the value of the type attribute of the configure XML element that is created.	`update`	No
`verbose`	Execute in verbose mode.	`false`	No
`easerverpropsonly`	Whether the generated XML file should include all properties, or only those properties that are not set in an equivalent J2EE deployment descriptor for the entity. Specify true to exclude properties that are set in the equivalent J2EE deployment descriptor.	`false`	No
entity	The name of the entity being created, in the form *EntityType:EntityName*.	-	Yes

The exportconfig command creates the file *sybase-easerver-config.xml* in the specified directory. The command fails if a file with this name already exists.

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example creates an XML configuration file in the current directory for the component *EventSamples/StockManager* and specifies `create` as the configure type:

```
jagtool exportconfig -configtype create Component:EventSamples/StockManager
```

flushstaticpage

Description

Flushes the static page cache.

Syntax

Local mode support: No.

Command line:

flushstaticpage *connect-args*

Ant build file:

<jag_flushstaticpage />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See “Using connected mode” on page 216.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

See also “Static Page Caching” on page 39.

gen_skels

Description Generates skeletons for a package or component.

Syntax **Local mode support:** Yes.

Command line:

gen_skels
[*connect-args* | *local-args*]
[-javaskelcodebase *path*]
[-cppskelcodebase *path*]
[-javastubtype *CORBA*|*EJB*|*Jaguar1.1*]
[-compilejavaskels *true*|*false*]
[-pseudoskels *true*|*false*]
[-jdkversion *1.1*|*1.2*]
entity

Ant build file:

<jag_gen_skels
[javaskelcodebase="*path*"]
[cppskelcodebase="*path*"]
[compilejavaskels="*true*|*false*"]
[pseudoskels="*true*|*false*"]
[javastubtype=”*CORBA*|*EJB*|*Jaguar1.1*”]

[jdkversion="*1.1*|*1.2*"]
entity="*entity*" />

Option	Description	Default	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	-	Yes
`javaskelcodebase`	The code base for Java skeletons, on the server host.	*$JAGUAR/java/classes*	No
`cppskelcodebase`	The code base for C++ skeletons, on the server host.	*$JAGUAR/cpplib*	No
`javastubtype`	The type of Java stubs used. Options are: • CORBA • EJB • Jaguar1.1	CORBA	No
`cppstubcodebase`	The code base for C++ stubs, on the server host.	*$JAGUAR/include*	No
`compilejavaskels`	Indicates whether to compile Java skeletons.	true	No
`pseudoskels`	Indicates whether to generate direct-access skeletons for C++ and Java/CORBA pseudocomponents. Use this option if the target components run as EAServer pseudocomponents.	false	No
`jdkversion`	Indicates the JDK version to use for compiling Java skeletons. Options are: • 1.1 – JDK 1.1 • 1.2 – JDK 1.2 or later	1.2	No
entity	The name of the entity for which skeletons are generated, in the form *EntityType:EntityName*, where *EntityType* must be either *Package* or *Component*.	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example generates skeletons with EJB-compatible Java stubs for Package:SVU:

- Command line:

```
jagtool gen_skels -javastubtype EJB Package:SVU
```

- Ant build file:

```
<jag_gen_skels javastubtype="EJB" entity="Package:SVU" />
```

See also gen_stubs, gen_stubsandskels

gen_stubs

Description Generates stubs for a package or component.

Syntax **Local mode support:** Yes.

Command line:

gen_stubs
[*connect-args* | *local-args*]
[-javastubs *true*|*false*]
[-cppstubs *true*|*false*]
[-javastubtype *CORBA*|*EJB*|*Jaguar1.1*]
[-javastubjarname *filename*]
[-javastubcodebase *path*]
[-cppstubcodebase *path*]
[-compilejavastubs *true*|*false*]
[-pseudostubs *true*|*false*]
[-jdkversion *1.1*|*1.2*]
entity

Ant build file:

<jag_gen_stubs
[javastubs="*true*|*false*"]
[cppstubs="*true*|*false*"]
[javastubtype="*CORBA*|*EJB*|*Jaguar1.1*"]
[javastubjarname="*filename*"]
[javastubcodebase="*path*"]
[cppstudcodebase="*path*"]
[pseudostubs="*true*|*false*"]
[compilejavastubs="*true*|*false*"]
[jdkversion="*1.1*|*1.2*"]
entity="*entity*" />

Option	Description	Default	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	-	Yes
`javastubs`	Indicates whether to generate Java stubs.	true	No

Option	Description	Default	Required
`cppstubs`	Indicates whether to generate C++ stubs.	true	No
`javastubstype`	The type of Java stubs used. Options are: • CORBA • EJB • Jaguar1.1	CORBA	No
`javastubjarname`	If this flag is used, the generated Java stubs are placed in a JAR file with the specified file name on the server host. The full path for the file name must be specified.	-	No
`javastubcodebase`	The code base for Java stubs, on the server host.	*$JAGUAR/html/classes*	No
`cppstubcodebase`	The code base for C++ stubs, on the server host.	*$JAGUAR/include*	No
`compilejavastubs`	Indicates whether to compile Java stubs.	true	No
`pseudostubs`	Indicates whether to generate direct-access stubs for C++ and Java/CORBA pseudocomponents. Use this option if the target components run as EAServer pseudocomponents.	false	No
`jdkversion`	Indicates the JDK version to use for compiling Java skeletons. Options are: • 1.1 – JDK 1.1 • 1.2 – JDK 1.2 or later	1.2	No
entity	The name of the entity for which stubs are generated, in the form *EntityType:EntityName*, where *EntityType* must be either *Package* or *Component*.	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example generates EJB stubs for Package:SVU. The stubs are automatically compiled and added to the JAR file *svustubs.jar*:

- Command line (all on one line):

```
jagtool gen_stubs -javastubtype EJB -javastubjarname
e:\temp\svustubs.jar Package:SVU
```

- Ant build file:

```
<jag_gen_stubs javastubtype="EJB"
javastubjarname="e:\temp\svustubs.jar" entity="Package:SVU" />
```

See also gen_skels, gen_stubsandskels

gen_stubsandskels

Description Generates stubs and skeletons for a package or a component.

Syntax **Local mode support:** Yes.

Command line:

gen_stubsandskels
[*connect-args* | *local-args*]
[-javastubs *true*|*false*]
[-cppstubs *true*|*false*]
[-javastubtype *CORBA*|*EJB*|*Jaguar1.1*]
[-javastubjarname *filename*]
[-javastubcodebase *path*]
[-cppstudcodebase *path*]
[-compilejavastubs *true*|*false*]
[-generatedependentclasses *true*|*false*]
[-skels *true*|*false*]
[-javaskelcodebase *path*]
[-cppskelcodebase *path*]
[-compilejavaskels *true*|*false*]
[-pseudostubsandskels *true*|*false*]
[-jdkversion *1.1*|*1.2*]
[-verbose *true*|*false*]
entity

Ant build file:

<jag_gen_stubsandskels
[javastubs="*true*|*false*"]
[cppstubs="*true*|*false*"]
[javastubtype="*CORBA*|*EJB*|*Jaguar1.1*"]
[javastubjarname="*filename*"]
[javastubcodebase="*path*"]
[cppstudcodebase="*path*"]
[compilejavastubs="*true*|*false*"]
[generatedependentclasses="*true*|*false*"]
[skels="*true*|*false*"]
[javaskelcodebase="*path*"]
[cppskelcodebase="*path*"]
[compilejavaskels="*true*|*false*"]

[pseudostubsandskels="*true*|*false*"]
[jdkversion="*1.1*|*1.2*"]
[verbose="*true*|*false*"]
entity="*entity*" />

Option	Description	Default	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	-	Yes
`javastubs`	Indicates whether to generate Java stubs.	true	No
`cppstubs`	Indicates whether to generate C++ stubs.	true	No
`javastubstype`	The type of Java stubs used. Options are: • CORBA • EJB • Jaguar1.1	CORBA	No
`javastubjarname`	If this flag is used, the generated Java stubs are placed in a JAR file with the specified file name on the server host. The full path for the file name must be specified.	-	No
`javastubcodebase`	The code base for Java stubs, on the server host.	*$JAGUAR/html/classes*	No
`cppstubcodebase`	The code base for C++ stubs, on the server host.	*$JAGUAR/include*	No
`compilejavastubs`	Indicates whether to compile Java stubs.	true	No
`generatedependent-classes`	For EJB components that have been deployed in an EJB-JAR file, specifies whether to include additional Java classes that are referenced by the bean's client interfaces but that are not included in the EJB-JAR file or from packages that start with java., javax. or com.sybase.ejb.	true	No
`skels`	Indicates whether to generate skeletons.	true	No
`javaskelcodebase`	The code base for Java skeletons, on the server host.	*$JAGUAR/java/classes*	No
`cppskelcodebase`	The code base for C++ skeletons, on the server host.	*$JAGUAR/cpplib*	No
`compilejavaskels`	Indicates whether to compile Java skeletons.	true	No
`pseudostubsandskels`	Indicates whether to generate direct-access stubs and skeletons for C++ and Java/CORBA pseudocomponents. Use this option if the target components run as EAServer pseudocomponents.	false	No

Option	Description	Default	Required
`jdkversion`	Indicates the JDK version to use for compiling Java skeletons. Options are: • 1.1 – JDK 1.1 • 1.2 – JDK 1.2 or later	1.2	No
`verbose`	Indicates whether to generate verbose output.	false	No
entity	The name of the entity for which stubs are generated, in the form *EntityType:EntityName*, where *EntityType* is either "Package" or "Component."	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example generates skeletons and EJB stubs for Package:SVU. The stubs and skeletons are automatically compiled.

- Command line:

```
jagtool gen_stubsandskels -javastubtype EJB Package:SVU
```

- Ant build file:

```
<jag_gen_stubsandskels="EJB" entity="Package:SVU" />
```

See also

gen_skels, gen_stubs

gen_tlbreg

Description

Generates the *.tlb* and *.reg* files that you need to run an ActiveX client.

This command can be run only on a Windows machine, and when the server to which jagtool connects is running in Windows.

Syntax

Local mode support: Yes.

Command line:

```
gen_tlbreg
[ connect-args | local-args ]
-proxyserverloc pspath
[-outputdir outdir]
[-register true|false]
[-savemidl true|false]
entity
```

Ant build file:

```
<jag_gen_tlbreg proxyserverloc="pspath"
[outputdir="outdir"]
[register="true|false"]
[savemidl="true|false"]
entity
```

Option	Description	Default	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	-	Yes
`proxyserverloc`	The path to the InProcServer corresponding to the Generic ActiveX Proxy DLL (*jagproxy.dll*). **Warning!** Do not leave this field blank. If you do, an empty string is inserted into the InProcServer32 entry in the Windows Registry and the ActiveX proxy does not run.	-	Yes
`outputdir`	The output directory to store the generated *.tlb* and *.reg* files.	Root directory on the drive where EAServer is installed	No
`register`	If true, registers the ActiveX Proxy interfaces.	false	No
`savemidl`	If true, the generated Microsoft interface definition language is retained.	false	No
entity	Name of the entity for which *.tlb* and *.reg* files are generated, in this form: *EntityType*:*EntityName* where *EntityType* is either "Package" or "Module."	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

See also

Chapter 20, “Creating ActiveX Clients,” in the *EAServer Programmer’s Guide*.

getmonitorstats

Description

Retrieves and prints runtime monitoring statistics from the server to which you are connected.

Syntax

Local mode support: No.

Command line:

getmonitorstats *connect-args item* [*detail*]

Ant build file:

<jag_getmonitorstats [item=”*item*”] [detail=”*detail*”] />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See “Using connected mode” on page 216.	Yes
item	An item type as listed in Table 12-4.	Yes
detail	An optional detail specifier to match the specified item type, as listed in Table 12-4.	No

Usage

getmonitorstats allows you to retrieve runtime monitoring statistics for the items listed in Table 12-4.

Table 12-4: getmonitorstats options

Item	Detail	To specify
Package	An optional package name.	Statistics for components in the specified package, or all components if you do not specify a package name.
Component	A component name in the form `package/component`.	Statistics for the specified component.
ConnCache	An optional connection cache name.	Statistics for the specified connection cache, or all of them if you do not specify a name.
ManagedConnectionFactory	An optional managed connection factory name.	Statistics for the specified managed connection factory, or all of them if you do not specify a name.
Network	The protocol, that is, `HTTP` or `IIOP`.	Network statistics for the specified protocol.

See also

Chapter 11, “Runtime Monitoring.”

getserverinfo

Description

Print status and version information for the server that you are connected to.

Syntax

Local mode support: No.

Command line:

getserverinfo *connect-args* [-version *true*|*false*] [-status *true*|*false*]

Ant build file:

<jag_getseverinfo [version="*true*|*false*"] [status="*true*|*false*"] />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See "Using connected mode" on page 216.	Yes
`version`	Whether to print the server version number.	No
`status`	Whether to print the server status, that is, whether the server is accepting regular client connections or in Admin mode.	No

See also

set_admin, set_ready, "Using Admin mode" on page 70

getservicestate

Description

Returns the state of service components executing in the server.

Syntax

Local mode support: No.

Command line:

getservicestate *connect-args servicename*

Ant build file:

<jag_getsevicestate servicename="*servicename* "

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See "Using connected mode" on page 216.	Yes

Option	Description	Required
servicename	The name of the service to query, or all to list the state of all services. jagtool fails with an error message if you specify the name of a service that is not installed or that does not implement the interface `CtsServices::ExtendedService`.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Usage

This command prints the service state, using the terms listed in Table 12-5.

Table 12-5: Service states

State	Description
`UNKNOWN`	The state is unknown because the service does not implement the `CtsServices::ExtendedService` interface.
`STARTING`	The service is starting. The start method has been called, but has not returned.
`STARTED`	The service is started, but not yet running. The start method has returned, but run has not been called.
`RUNNING`	The service is running, that is, executing the run method.
`FINISHED`	The service is finished processing. The run method has returned. This state applies only to services that do not run continuously until stopped.
`STOPPING`	The service is stopping. The stop method has been called, and is still running.
`STOPPED`	The service is stopped. The stop method has been called and has returned.

For example, this is the typical output for a server where the message service is not installed:

```
jagtool getservicestate all
JaguarServlet/ServletService's state is RUNNING
CosNaming/JNameService's state is FINISHED
```

This command is useful when running jagtool or jagant from scripts that start or restart the server. You can check the output to determine if the required service is running. For example, components should not be deployed before the name service has finished, and JMS entities cannot be created before the message service is running.

See also

Chapter 33, "Creating Service Components," in the *EAServer Programmer's Guide*.

grantroleauth

Description

Grants authorization to a given role to perform specific actions on the given entity. If the entity is a server, members of the role are granted permission to restart, refresh, or shutdown the server. If the entity is an application, Web application, servlet, or package, members of the role are granted access to those resources, including deploying the entity.

Syntax

Local mode support: Yes.

Command line:

grantroleauth [*connect-args* | *local-args*] [-role *rolename*] \
[-action *actionname*] *entity*

Ant build file:

<jag_grantroleauth [role="*rolename*"] [action="*actionname*"] entity="*entity*" />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
`role`	The role id or name to which authorization is being granted. The role must exist on the server to which you are connected.	Yes
`action`	Only valid when the entity type is Server. Valid actions include restart, refresh and shutdown.	When the entity type is server.
entity	The name of the entity, in the form *EntityType:EntityName*. Valid entities are application, Webapplication, servlet, server, and package.	Yes

Examples

This example grants access to the “Estore” application to members of the role named “test”.

```
jagtool grantroleauth -role test Application:Estore
```

See also

removeroleauth

install

Description

Installs an entity into another entity (for example, installing a package into a server).

Syntax

Local mode support: Yes.

Command line:

install [*connect-args* | *local-args*] *source target*

Ant build file:

<jag_install source=”*source*” target=”*target*” />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	Yes
source	The entity identifier for the entity being installed.	Yes
target	The entity identifier in which the source entity is being installed.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example installs the package *SoapDemo* in the server.

- Command line:

```
jagtool install Package:SoapDemo Server:Jaguar
```

- Ant build file:

```
<jag_install source="Package:SoapDemo" target="Server:Jaguar" />
```

Example 2 This example installs the application *MyPortfolio* in the server.

- Command line:

```
jagtool install Application:MyPortfolio Server:Jaguar
```

- Ant build file:

```
<jag_install source="Application:MyPortfolio" target="Server:Jaguar" />
```

Example 3 This example installs *MyListener* into the Jaguar server:

- Command line:

```
jagtool install Listener:Jaguar/MyListener Server:Jaguar
```

- Ant build file:

```
<jag_install source="Listener:Jaguar/MyListener" target=”Server:Jaguar”
/>
```

Example 4 This example installs the service component *MyPack/MyComp* into the Jaguar server:

- Command line:

```
jagtool install Service:MyPack/MyComp Server:Jaguar
```

- Ant build file:

```
<jag_install source="Service:MyPack/MyComp" target=”Server:Jaguar” />
```

See also create, remove

jmscreate

Description Creates a JMS entity.

Syntax **Local mode support:** No.

Command line:

jmscreate *connect-args entity* [*file*]

Ant build file:

<jag_jmscreate entity=”*entity*” [file=”*file*”] />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See "Using connected mode" on page 216.	Yes
entity	The entity to create, in the form *EntityType*:*EntityName*. Valid entity types are: • MessageQueue • MessageTopic • QueueConnectionFactory • ThreadPool • TopicConnectionFactory	Yes
file	An optional file containing properties for the entity. The file must specify properties in the form of an EAServer repository properties file. You can set properties after the entity exists with the jmsset_props command.	No

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example uses the command line to create a message queue named MyQueue and specifies a properties file:

```
jagtool jmscreate MessageQueue:MyQueue
D:\Jag41005\sample\jagtool\queueprops.txt
```

Example 2 This example does the same thing in an Ant build file:

```
<jag_jmscreate entity="MessageQueue:AntQueue"
file="D:\Jag41005\sample\jagtool\queueprops.txt" />
```

See also

jmsdelete, jmslist, jmsprops, jmsset_props

"Permanent destinations" on page 165 describes properties for message queues and message topics.

"Connection factories" on page 168 describes properties for queue connection factories and topic connection factories.

"Thread pools" on page 174 describes thread pool properties.

jmsdelete

Description

Deletes the specified entity.

Syntax

Local mode support: No.

Command line:

jmsdelete *connect-args entity*

Ant build file:

<jag_jmsdelete entity="*entity*" />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See "Using connected mode" on page 216.	Yes
entity	The entity to delete, in the form *EntityType*:*EntityName*. Valid entity types are: • Listener • MessageQueue • MessageTopic • QueueConnectionFactory • ThreadPool • TopicConnectionFactory	Yes

If the entity is a message queue, you must first remove its listeners. This operation removes selectors, and receives and acknowledges all queued messages for the message queue.

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This command line example deletes the message queue "MyQueue":

```
jagtool jmsdelete MessageQueue:MyQueue
```

Example 2 This example does the same thing in an Ant build file:

```
<jag_jmsdelete entity="MessageQueue:MyQueue" />
```

jmsflush

Description

Flushes the messages from the specified message queue.

Syntax

Local mode support: No.

Command line:

jmsflush *connect-args* MessageQueue:*QueueName*

Where *connect-args* is the list of arguments to specify a connection to the server. See “Using connected mode” on page 216.

Ant build file:

<jag_jmsflush entity=”MessageQueue:*QueueName*” />

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This command line example flushes the message queue “MyQueue”:

```
jagtool jmsflush MessageQueue:MyQueue
```

Example 2 This example does the same thing in an Ant build file:

```
<jag_jmsflush entity="MessageQueue:MyQueue" />
```

jmslist

Description

Lists JMS entities of the specified type.

Syntax

Local mode support: No.

Command line:

jmslist *connect-args type*

Ant build file:

<jag_jmslist type=”*type*” />

Where *connect-args* is the list of arguments to specify a connection to the server, described in “Using connected mode” on page 216, and *type* is one of the following JMS entity types:

Type specifier	To list names of
`ActiveMessageQueue`	Currently active message queues.
`ConfMessageQueue`	Configured message queues (queues that were created administratively rather than programmatically).
`ActiveMessageTopic`	Currently active message topics.
`ConfMessageTopic`	Configured message topics (topics that were created administratively rather than programmatically).
`QueueConnectionFactory`	Queue connection factories.
`ThreadPool`	Thread pools.
`TopicConnectionFactory`	Topic connection factories.

<system> message queue and thread pool The message service includes a message queue and a thread pool called "<system>" for tasks that require internal messaging, such as synchronizing a cluster. The <system> message queue and thread pool are visible in EAServer Manager and as output from jagtool jmslist commands.

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This command line example lists active JMS message queues:

```
jagtool jmslist ActiveMessageQueue
```

Example 2 This example does the same in an Ant build file:

```
<jag_jmslist type="ActiveMessageQueue" />
```

jmslist_listeners

Description

Lists the names of the listeners attached to the specified message queue.

Syntax

Local mode support: No.

Command line:

jmslist_listeners *connect-args* MessageQueue:*QueueName*

Where *connect-args* is the list of arguments to specify a connection to the server. See “Using connected mode” on page 216.

Ant build file:

<jag_jmslist_listeners type=”MessageQueue:*QueueName*” />

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This command line example lists the listeners attached to “MyQueue”:

```
jagtool jmslist_listeners MessageQueue:MyQueue
```

Example 2 This example does the same in an Ant build file:

```
<jag_jmslist_listeners type="MessageQueue:MyQueue"
/>
```

jmslist_messages

Description

Lists the messages in the specified message queue.

Syntax

Local mode support: No.

Command line:

jmslist_messages *connect-args* [-maximum #*messages*]
[-selector *expression*]
MessageQueue:*QueueName*

Ant build file:

<jag_jmslist_messages [“maximum = #*messages*]
[“selector = *expression*”]
type=”MessageQueue:*QueueName*” />

Option	Description	Default	Required
connect-args	Arguments to specify a connection to the server. See “Using connected mode” on page 216.	-	Yes

Option	Description	Default	Required
#messages	The maximum number of messages to list. If 0 or a negative number, all messages are listed.	100	No
expression	A selector expression. Only messages that match the selector expression are returned. If set to true, all messages are returned. The number of these messages that are listed is determined by *#messages*.	true	No

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This command line example lists the first 25 messages in "MyQueue":

```
jagtool jmslist_messages -maximum 25 MessageQueue:MyQueue
```

Example 2 This example does the same in an Ant build file:

```
<jag_jmslist_messages "maximum=25" type="MessageQueue:MyQueue" />
```

jmsmanage_listeners

Description

Adds and removes listeners to and from JMS message queues.

Syntax

Local mode support: No.

Command line:

jmsmanage_listeners
connect-args
-action "Component:*comp*"
MessageQueue:*queue*

Ant build file: <

jag_manage_listeners action="*action*" listener="Component:*comp*"
entity="MessageQueue:*queue*"/>

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See "Using connected mode" on page 216.	Yes
action	`add` to add the listener to the queue or `remove` to remove the listener from the queue.	Yes
comp	The component that listens for messages on the specified queue, in the format *`package/component`*.	Yes
queue	The message queue name.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

These commands add and remove the component JmsListenerTest/JmsListener to the queue MyQueue:

```
jagtool jmsmanage_listeners -add Component:JmsListenerTest/JmsListener
MessageQueue:MyQueue

jagtool jmsmanage_listeners -remove Component:JmsListenerTest/JmsListener
MessageQueue:MyQueue
```

jmsmanage_selectors

Description

Adds and removes selectors to and from JMS message queues.

Syntax

Local mode support: No.

Command line:

jmsmanage_selectors
connect-args
-action "*selector*"
MessageQueue:*queue*

Ant build file:

<jag_manage_selectors action="*action*" selector="*selector*"
entity="MessageQueue:*queue*"/>

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See “Using connected mode” on page 216.	Yes
action	`add` to add the selector to the queue or `remove` to remove the selector from the queue.	Yes
selector	The JMS selector.	Yes
queue	The message queue name.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

These commands install and remove a selector `topic='test'` from the queue MyQueue:

```
jagtool jmsmanage_selectors -add "topic='test'" MessageQueue:MyQueue

jagtool jmsmanage_listeners -remove "topic='test'" MessageQueue:MyQueue
```

jmsprops

Description

Lists the properties for a JMS entity.

Syntax

Local mode support: No.

Command line:

jmsprops *connect-args entity*

Ant build file:

<jag_jmsprops entity=”*entity*” />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See “Using connected mode” on page 216.	Yes

Option	Description	Required
entity	The entity of interest, in the form *EntityType*:*EntityName*. Valid entity types are: • MessageQueue • MessageTopic • QueueConnectionFactory • ThreadPool • TopicConnectionFactory	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example lists properties for the message queue "MyQueue":

```
jagtool jmsprops MessageQueue:MyQueue
```

Example 2 This example does the same thing in an Ant build file:

```
<jag_jmsprops entity="MessageQueue:MyQueue" />
```

See also

jmsset_props

jmsset_props

Description

Sets properties for a JMS entity.

Syntax

Local mode support: No.

Command line: To set an individual property, specify the property name and value on the command line:

jmsset_props *connect-args entity name value*

To set multiple properties (one or more) from a properties file, specify the file name:

jmsset_props *connect-args entity file*

Ant build file: To set an individual property, specify the property name and value:

<jag_jmsset_props entity="*entity*" name="*name*" value="*value*" />

To set multiple properties (one or more) from a properties file, specify the file name:

<jag_jmsset_props entity="*entity*" file="*file*" />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See "Using connected mode" on page 216.	Yes
entity	The entity of interest, in the form *EntityType*:*EntityName*. Valid entity types are: • MessageQueue • MessageTopic • QueueConnectionFactory • ThreadPool • TopicConnectionFactory	Yes
name	The property name.	No
value	The property value.	No
file	An optional file containing properties for the entity. The file must specify properties in the form of an EAServer repository properties file.	No

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This command line example configures the "maximum" property of the queue named "AntQueue":

```
jagtool jmsset_props "MessageQueue:AntQueue" maximum 99
```

This example does the same thing in an Ant build file:

```
<jag_jmsset_props entity="MessageQueue:AntQueue" name="maximum" value="99"
/>
```

Example 2 This command line example configures the queue named "AntQueue," specifying a properties file:

```
jagtool jmsset_props "MessageQueue:AntQueue"
"D:\Jag41005\sample\jagtool\Newqueueprops.txt"
```

This example does the same thing in an Ant build file:

```
<jag_jmsset_props entity="MessageQueue:AntQueue"
file="D:\Jag41005\sample\jagtool\Newqueueprops.txt" />
```

Example 3 Here is what the *Newqueueprops.txt* file used in the above examples might contain. See “Permanent destinations” on page 165 for an explanation of these properties:

```
IGNORE_DUPLICATE_KEY=false
REQUIRES_ACKNOWLEDGE=false
REQUIRES_TRANSACTION=false
maximum=0
qop=none
share=true
store=true
table=
timeout=60
```

See also

jmscreate, jmsdelete, jmslist, jmsprops

“Permanent destinations” on page 165 describes properties for message queues and message topics.

“Connection factories” on page 168 describes properties for queue connection factories and topic connection factories.

“Thread pools” on page 174 describes thread pool properties.

list

Description

Lists entities in the repository.

Syntax

Local mode support: Yes.

Command line:

list [*connect-args* | *local-args*] *type entity*

Ant build file:

<jag_list type=”*type*” entity=”*entity*” />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	Yes

Option	Description	Required
type	The type of entities to list.	Either *type* or *entity* is required. Both can be used.
entity	An optional entity identifier to specify a parent entity. Child entities are listed.	

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This command lists all connection caches, running from the command line in local mode:

```
jagtool -local list ConnCache
```

Example 2 This example lists all the packages in the repository.

- Command line:

```
jagtool list Package
```

- Ant build file:

```
<jag_list type="Package" />
```

Example 3 This example lists all the child entities of Package:SVU.

- Command line:

```
jagtool list Package:SVU
```

- Ant build file:

```
<jag_list entity="Package:SVU" />
```

Example 4 This example lists all the child components of Package:SVU.

- Command line:

```
jagtool list Component Package:SVU
```

- Ant build file:

```
<jag_list type="Component" entity="Package:SVU" />
```

list_ver

Description

Lists the versions of an entity.

Syntax

Local mode support: Yes.

Command line:

list_ver [*connect-args* | *local-args*] *entity*

Ant build file:

<jag_list_ver entity="*entity*" />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
entity	The name of the entity for which versions are listed, in the form *EntityType*:*EntityName*.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

This example lists versions of Package:SVU.

- Command line:

```
jagtool list_ver Package:SVU
```

- Ant build file:

```
<jag_list_ver entity="Package:SVU" />
```

See also

restore_ver, save_major_ver, save_minor_ver, Chapter 10, "Using Repository Versioning"

merge_props

Description

Merges or deletes property values for an entity.

Syntax

Local mode support: Yes.

Command line:

```
merge_props [ connect-args | local-args ] entity \
   [-verbose true|false] { [mergeop name value] | [file] }
```

Ant build file: There are three syntax forms for Ant commands. You can specify a merge command for a single property with this syntax:

```
<jag_merge_props entity="entity" [ verbose=" true|false" ]
mergeop="mergeop" name="name" value="value" >
</jag_merge_props>
```

You can specify merge commands for multiple properties with this syntax:

```
<jag_merge_props entity="entity" [ verbose="true|false" ]>
  <mergeproperty mergeop="mergeop" name="name"
    value="value" />
  <mergeproperty mergeop="mergeop" name="name"
    value="value" />
  ...
</jag_merge_props>
```

You can specify the name of a file that contains merge commands with this syntax:

```
<jag_merge_props entity="entity" [ verbose=" true|false" ]
  file="file" ></jag_merge_props>
```

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
entity	The entity identifier for the target entity. Valid entity types include Agent, Application, Cluster, Component, ConnCache, Connector, DatabaseType, EntityCollection, Filter, InstancePool, Listener, ManagedConnectionFactory, Package, Role, Security, Server, Servlet, and WebApplication. The entity types Connector, Filter, and ManagedConnectionFactory are available only for J2EE 1.3-enabled servers.	Yes
`verbose`	Whether to execute in verbose mode. The default is `false`.	No
mergeop	The merge operation, one of: • **AppendToList** For properties that take a comma-separated list of values, append the value to the existing value if not already present. Case matters when comparing against existing entries; for example, if foo2 is present and you append Foo2, both are present after appending. • **PrependToList** For properties that take a comma-separated list of values, prepend the value to the existing value. • **RemoveFromList** For properties that take a comma-separated list of values, remove the value from the existing value if present. • **SetDefault** Set the properties value to the default value. • **Delete** Delete the property setting completely.	When not specifying a file containing merge commands.

Option	Description	Required
name	The name of the property of interest.	When not specifying a file containing merge commands.
value	The property value to merge.	When not specifying a file containing merge commands.
file	The name of a text file containing merge commands. The file must be a text file containing lines of the form: *mergeop*:*name*=*value* Where *mergeop*, *name*, and *value* follow the syntax rules above.	When not specifying a merge operation and property name.

Examples

Example 1 To add a service to the list of services for server "Jaguar," you can run the following on the command line:

```
jagtool merge_props Server:Jaguar AppendToList
com.sybase.jaguar.server.services MyNewService
```

Example 2 The following Ant example appends a value to the Java classes setting for a package:

```
<target name="test_merge_props" depends="connect">
<jag_merge_props entity="Package:Foo"
  verbose="true" mergeop="AppendToList"
  name="com.sybase.jaguar.package.java.classes"
  value="com.foo.MyClass">
</jag_merge_props>
</target>
```

Example 3 This Ant example merges several properties for a package:

```
<target name="test_merge_props" depends="connect">
<jag_merge_props entity="Package:Foo" verbose="true">
  <mergeproperty mergeop="AppendToList"
    name="com.sybase.jaguar.package.description" value="more stuff"/>
  <mergeproperty mergeop="PrependToList"
    name="com.sybase.jaguar.package.someprop" value="newvalue3"/>
  <mergeproperty mergeop="RemoveFromList"
    name="com.sybase.jaguar.package.someprop" value="original"/>
  <mergeproperty mergeop="SetDefault"
    name="com.sybase.jaguar.package.someprop" value="newvalue4"/>
  <mergeproperty mergeop="Delete"
```

```
    name="com.sybase.jaguar.package.someprop"/>
</jag_merge_props>
</target>
```

Example 4 This Ant example specifies the name of a file that contains merge commands:

```
<target name="test_merge_props" depends="connect">
<jag_merge_props entity="Package:Foo"
  verbose="true" file="C:\EAServer\MergeProps.txt">
</jag_merge_props>
</target>
```

See also

props, set_props

ping

Description

Pings a connection cache.

Syntax

Local mode support: No.

Command line:

ping *connect-args entity*

Ant build file:

<jag_ping entity="*entity*" />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See "Using connected mode" on page 216.	Yes
entity	The connection cache identifier in the form `ConnCache:EntityName`.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example pings the connection cache named "JavaCache":

- Command line:

```
jagtool ping ConnCache:JavaCache
```

- Ant build file:

```
<jag_ping entity="ConnCache:JavaCache" />
```

props

Description

Lists properties for an entity in the repository.

Syntax

Local mode support: Yes.

Command line:

props [*connect-args* | *local-args*] *entity*

Ant build file:

<jag_props entity="*entity*" />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
entity	The entity identifier for the entity whose properties are listed.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example lists the properties of Package:SVU.

- Command line:

```
jagtool props Package:SVU
```

- Ant build file:

```
<jag_props entity="Package:SVU" />
```

Example 2 This example lists the properties of Server:Jaguar.

- Command line:

```
jagtool props Server:Jaguar
```

- Ant build file:

```
<jag_props entity="Server:Jaguar" />
```

See also — Appendix B, “Repository Properties Reference”

rebind

Description — Rebinds a cluster, which refreshes all of the name servers within the cluster.

Syntax

Local mode support: No.

Command line:

rebind *connect-args* Cluster:*name*

Ant build file:

<jag_rebind entity=”Cluster:*name*” />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See “Using connected mode” on page 216.	Yes
name	The name of the cluster to rebind.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Usage — If you add a component to a server that is already part of a cluster and want to make that component available to the cluster, you need to rebind the cluster. You can also use the rebind option if a problem occurs when you synchronize the cluster; if for example, one of the name servers is slow to start.

See also — Chapter 6, “Clusters and Synchronization.”

refresh

Description

Refreshes an entity in the server.

Syntax

Local mode support: No.

Command line:

refresh *connect-args entity*

Ant build file:

<jag_refresh entity="*entity*" />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See "Using connected mode" on page 216.	Yes
entity	The entity identifier for the entity being refreshed. To refresh servers, you must be connected to the specified server. To refresh other entities, you must be connected to a server in which the entity is installed. Refresh does not support servlets. To refresh servlets, refresh the Web application or server in which they are installed.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example refreshes Package:SVU.

- Command line:

```
jagtool refresh Package:SVU
```

- Ant build file:

```
<jag_refresh entity="Package:SVU" />
```

Example 2 This example refreshes the SVULogin component of Package:SVU.

- Command line:

```
jagtool refresh Component:SVU/SVULogin
```

- Ant build file:

```
<jag_refresh entity="Component:SVU/SVULogin" />
```

Example 3 This example refreshes the Jaguar server, and works only when you are connected to the server with this name.

- Command line:

```
jagtool refresh Server:Jaguar
```

- Ant build file:

```
<jag_refresh entity="Server:Jaguar" />
```

remove

Description

Removes, but does not delete, an entity from another entity. For example, use remove to remove a package from a server.

Syntax

Local mode support: Yes.

Command line:

remove [*connect-args* | *local-args*] *source target*

Ant build file:

<jag_remove source="*source*" target="*target*" />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
source	The entity identifier of the entity being removed.	Yes
target	The entity identifier of the entity from which the *source* is removed.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example removes Package:SVU from the entity Server:Jaguar.

- Command line:

```
jagtool remove Package:SVU Server:Jaguar
```

- Ant build file:

```
<jag_remove source="Package:SVU" target="Server:Jaguar"/>
```

Example 2 This example removes WebApplication:WebTier from the entity Application:estore.

- Command line:

```
jagtool remove WebApplication:WebTier Application:estore
```

- Ant build file:

```
<jag_remove source="WebApplication:WebTier" target="Application:estore"
/>
```

Example 3 This example removes the service component *MyPack/MyComp* from the Jaguar server:

- Command line:

```
jagtool remove Service:MyPack/MyComp Server:Jaguar
```

- Ant build file:

```
<jag_remove source="Service:MyPack/MyComp" target=”Server:Jaguar” />
```

See also install

removeroleauth

Description Removes authorization from members of a given role to perform specific actions on the given entity. If the entity is a server, members of the role are denied permission to restart, refresh, or shut down the server. If the entity is an application, Web application, servlet, or package, members of the role are denied access to those resources, including deploying the entity.

Syntax **Local mode support:** Yes.

Command line:

removeroleauth [*connect-args* | *local-args*] [-role *rolename*] \
[-action *actionname*] *entity*

Ant build file:

<jag_removeroleauth [role="*rolename*"] [action="*actionname*"] entity="*entity*" />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
`role`	Role id or name to which authorization is being denied. The role must exist on the server to which you are connected.	Yes
`action`	Only valid when the entity type is Server. Valid actions include restart, refresh and shutdown.	No
entity	The name of the entity, in the form *EntityType:EntityName*. Valid entities are application, Webapplication, servlet, server, and package.	Yes

Examples

This example denies access to the "Estore" application to members of the role named "test".

```
jagtool removeroleauth -role test Application:Estore
```

See also

grantroleauth

resref

Description

Sets the value of a J2EE resource reference.

Syntax

Local mode support: Yes.

Command line:

resref [*connect-args* | *local-args*] *entity* -refname *name* -value *value*

Ant build file:

<jag_resref entity="*entity*" refname="*name*" value="*value*" />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See "Local versus connected mode" on page 216.	Yes
entity	The entity identifier in the form *EntityType*:*EntityName*.	Yes

Option	Description	Required
name	The name of a resource reference for the given entity.	Yes
value	The value for the reference.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

This example sets the value of a reference in the component TheAccount in the Customer_Component package. The resource reference jdbc/EstoreDataSource is set to the value "PetStoreDB."

- Command line (all on one line):

```
jagtool resref Component:Customer_Component/TheAccount -refname
jdbc/EstoreDataSource -value PetStoreDB
```

- Ant build file:

```
<jag_resref entity="Component:Customer_Component/TheAccount"
refname="jdbc/EstoreDataSource" value="PetStoreDB" />
```

restart

Description

Terminates and restarts the server process to which you are connected.

Syntax

Local mode support: No.

Command line:

restart *connect-args*

Where *connect-args* is a list of arguments to specify a server connection, as described in "Using connected mode" on page 216.

Ant build file:

<jag_restart />

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.

Return value	Indicates
2	The command did not run successfully; an exception was thrown.

Examples

This example connects to the server “eclipse” on port 9005, with the user name “jagadmin” and the password “jagpass,” and restarts the server.

- Command line:

```
jagtool -h eclipse -n 9005 -u jagadmin -p jagpass restart
```

- Ant build file:

```
<jag_connect host="eclipse" port="9005" user="jagadmin"
password="jagpass" />
<jag_restart />
```

All jagant commands depend on jag_connect. See “Using the jag_connect command” on page 223 for more information about jag_connect.

See also

shutdown

restore_ver

Description

Restores a version of an entity.

Syntax

Local mode support: Yes.

Command line:

restore_ver [*connect-args* | *local-args*] *version entity*

Ant build file:

<jag_restore_ver version=”*version*” entity=”*entity*” />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	Yes
version	The version number of the version to restore.	Yes
entity	The name of the entity for which a version is restored, in the form *EntityType*:*EntityName*.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example restores version 1.0 of Package:SVU.

- Command line:

```
jagtool restore_ver 1.0 Package:SVU
```

- Ant build file:

```
<jag_restore_ver version="1.0" entity="Package:SVU" />
```

See also

list_ver, save_major_ver, save_minor_ver, Chapter 10, “Using Repository Versioning”

save_major_ver

Description

Creates a major version of an entity.

Syntax

Local mode support: Yes.

Command line:

save_major_ver [*connect-args* | *local-args*] [-c *comment*] *entity*

Ant build file:

<jag_save_major_ver [comment=”*comment*”] entity=”*entity*” />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	Yes
comment	A comment that describes the version.	No
entity	The name of the entity for which a version is created, in the form *EntityType*:*EntityName*.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.

Return value	Indicates
2	The command did not run successfully; an exception was thrown.

Examples

This example creates a major version of Package:SVU.

- Command line:

```
jagtool save_major_ver Package:SVU
```

- Ant build file:

```
<jag_save_major_ver comment="comment" entity="Package:SVU" />
```

See also

list_ver, restore_ver, save_minor_ver, Chapter 10, “Using Repository Versioning”

save_minor_ver

Description

Creates a minor version of an entity.

Syntax

Local mode support: Yes.

Command line:

save_minor_ver [*connect-args* | *local-args*] [-c *comment*] *entity*

Ant build file:

<jag_save_minor_ver [comment=”*comment*”] entity=“*entity*” />

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	Yes
comment	A comment that describes the version.	No
entity	The name of the entity for which a version is created, in the form *EntityType*:*EntityName*.	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

This example creates a minor version of Package:SVU.

- Command line:

```
jagtool save_minor_ver Package:SVU
```

- Ant build file:

```
<jag_save_minor_ver comment="comment" entity="Package:SVU" />
```

See also

list_ver, restore_ver, save_major_ver, Chapter 10, “Using Repository Versioning”

set_admin

Description

Puts the server that you have connected to into admin mode. The server must be restarted for the change to take effect.

Syntax

Local mode support: No.

Command line:

set_admin *connect-args* [*reason*]

Ant build file:

<jag_set_admin [reason=“*reason*”] />

Option	Description	Required
connect-args	Arguments to specify a connection to the server. See “Using connected mode” on page 216.	Yes
reason	The reason the server is being put into admin mode.	No

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example puts the server into admin mode with the reason “Regularly scheduled maintenance”.

- Command line:

```
jagtool set_admin "Regularly scheduled maintenance"
```

- Ant build file:

```
<jag_set_admin reason="Regularly scheduled maintenance" />
```

See also restart, set_ready, “Using Admin mode” on page 70

set_props

Description Sets properties for an entity in the repository. Properties can be set by specifying either a names and values or a properties file.

Syntax **Local mode support:** Yes.

Command line:

set_props [*connect-args* | *local-args*] *entity* [*name value*] | [*file*]

Ant build file, specifying a property file to read:

<jag_set_props entity=”*entity*” file=”*file*” />

Ant build file, specifying properties directly:

<jag_set_props entity=”*entity*” />
<property name=”*name*” value=”*value*”>
...
</jag_set_props>

Option	Description	Required
connect-args | *local-args*	Arguments to specify a connection to the server or to run in local mode. See “Local versus connected mode” on page 216.	Yes
entity	The entity identifier for the entity whose properties are being set.	Always.
name	The property name. In an Ant build file, you may specify multiple properties as `<property>` elements.	When setting properties directly.
value	The property value.	When setting properties directly.
file	The name of a property file. Files must specify properties in the format of an EAServer properties file.	When setting properties using a properties file.

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

Example 1 This example sets the description for the entity Package:SVU.

- Command line:

```
jagtool set_props Package:SVU com.sybase.jaguar.description "This is the
SVU Package"
```

- Ant build file:

```
<jag_set_props entity="Package:SVU">
<property name="com.sybase.jaguar.description" value="This is the SVU
Package" />
</jag_set_props>
```

Example 2 This example sets the properties for Package:SVU from the file *SVU.props*.

- Command line:

```
jagtool set_props Package:SVU SVU.props
```

- Ant build file:

```
<jag_set_props entity="Package:SVU" file="SVU.props" />
```

Example 3 You can use the Ant build file to specify multiple properties. For example, this declaration sets the values for the com.sybase.jaguar.description and com.sybase.jaguar.package.roles properties for Package:SVU.

- Ant build file:

```
<jag_set_props entity="Package:SVU" />
<property name="com.sybase.jaguar.description" value="This is the SVU
Package" />
<property name="com.sybase.jaguar.package.roles" value="jagadmin,role1"
/>
</jag_set_props>
```

Example 4 This example sets the host, port, and network protocol values for "MyListener":

- Command line:

```
jagtool set_props Listener:Jaguar/MyListener
```

```
com.sybase.jaguar.listener.host victor
jagtool set_props Listener:Jaguar/MyListener
com.sybase.jaguar.listener.port 9050
jagtool set_props Listener:Jaguar/MyListener
com.sybase.jaguar.listener.protocol iiop
```

- Ant build file:

```
<jag_set_props entity="Listener:Jaguar/MyListener" />
   <property name="com.sybase.jaguar.listener.host" value="victor">
   <property name="com.sybase.jaguar.listener.port" value="9050">
   <property name="com.sybase.jaguar.listener.protocol" value="iiop">
</jag_set_props>
```

Example 5 This jagtool example shows how special characters can be escaped when running commands in DOS or Windows. In this case, the = in the value set must be escaped by quoting:

```
jagtool set_props WebApplication:onepage
com.sybase.jaguar.webapplication.session-config=(session-timeout"="30)
```

This syntax is equivalent:

```
jagtool set_props WebApplication:onepage
com.sybase.jaguar.webapplication.session-config="(session-timeout=30)"
```

See also Appendix B, “Repository Properties Reference”

set_ready

Description Puts the server that you have connected to into ready mode.

Syntax **Local mode support:** No.

Command line:

set_ready *connect-args*

Where *connect-args* is a list of arguments to connect to the server, as specified in “Using connected mode” on page 216.

Ant build file:

<jag_set_ready />

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example puts the server into ready mode after being in admin mode.

- Command line:

```
jagtool set_ready
```

- Ant build file:

```
<jag_set_ready />
```

See also

set_admin, “Using Admin mode” on page 70

setjagadminpasswd

Description

Changes the jagadmin password for a server.

Syntax

Local mode support: No.

Command line:

setjagadminpasswd *connect-args* [-server *ServerName*] *oldpassword newpassword*

Ant build file:

<jag_setjagadminpasswd [serverName=”*ServerName*”] oldpassword=”*oldpassword* “ newpassword=“*newpassword*” />

Option	Description	Default	Required
connect-args	Arguments to specify a connection to the server. See “Using connected mode” on page 216.	-	Yes
server	Indicates the server for which to set the jagadmin password. The specified server must be defined in the repository used by the server to which you are connected.	Jaguar	No
oldpassword	The current password for the server.	-	Yes

Option	Description	Default	Required
newpassword	The new password for the server.	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
1	The command ran successfully; the result is false/failure.
2	The command did not run successfully; an exception was thrown.

Examples

This example changes the jagadmin password on MyServer from “secret” to “newsecret”. The server MyServer must be in the same repository as the server to which jagtool connects.

- Command line:

```
jagtool setjagadminpasswd -server MyServer secret newsecret
```

- Ant build file:

```
<jag_setjagadminpasswd serverName="MyServer"
oldpassword="secret" newpassword="newsecret"/>
```

Usage

The server must be restarted before clients (including jagtool) can connect with the new password.

See also

restart

shutdown

Description

Shuts down the server to which you are connected.

Syntax

Local mode support: No.

Command line:

shutdown *connect-args*

Where connect-args is a list of arguments to connect to the server, as described in “Using connected mode” on page 216.

Ant build file:

<jag_shutdown />

Like all commands, shutdown requires connection flags at the command line and the jag_connect command in Ant build files (see "Using the jag_connect command" on page 223). shutdown terminates the server process that you have connected to.

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

This example connects to the server *eclipse* on port *9005*, with the user name *jagadmin* and the password *jagpass*, and shuts down the server.

- Command line:

```
jagtool -h eclipse -n 9005 -u jagadmin -p jagpass shutdown
```

- Ant build file:

```
<jag_connect host="eclipse" port="9005" user="jagadmin"
password="jagpass" />
<jag_shutdown />
```

See also

restart

sync

Description

Synchronizes entities in the current repository to one or more remote repositories. Synchronization can be used to create identically configured servers in a cluster, or to copy entities from one server to another.

Syntax

Local mode support: No.

Command line:

sync *connect-args*
[-clusterfiles *true*|*false*]
[-packagefiles *true*|*false*]
[-servletfiles *true*|*false*]
[-webappfiles *true*|*false*]
[-appfiles *true*|*false*]
[-clientappfiles *true*|*false*]
[-connectorfiles *true*|*false*]
[-syncwebappjavaclasses *true*|*false*]
[-newprimary *true*|*false*]

[-newversion *true*|*false*]
[-refresh *true*|*false*]
[-refresh *true*|*false*]
[-restart *true*|*false*]
[-waitfor *waittime*]
[-verbose *true*|*false*]
[-cluster *clustername*]
[-servers *serverURLS*] *entity*

Ant build file:

<jag_sync
[clusterfiles="*true*|*false*="]
[packagefiles="*true*|*false*="]
[servletfiles="*true*|*false*="]
[webappfiles="*true*|*false*="]
[appfiles="*true*|*false*"]
[clientappfiles="*true*|*false*"]
[connectorfiles="*true*|*false*"]
[syncwebappjavaclasses="*true*|*false*"]
[newprimary="*true*|*false*"]
[newversion="*true*|*false*"]
[refresh="*true*|*false*"]
[refresh="*true*|*false*"]
[restart="*true*|*false*"]
[waitfor="*waittime*"]
[verbose="*true*|*false*"]
[cluster="*clustername*"]
[servers="*serverURLS*"]
entity="*entity*" />

Option	Description	Default	Required
connect-args	Arguments to specify a connection to the server. See “Using connected mode” on page 216.	-	Yes
`clusterfiles`	Indicates all cluster files should be synchronized (all files in Repository/Security, the property file for the cluster, and any server properties found in the cluster definition).	false	No
`packagefiles`	Indicates all packages should be synchronized regardless of the type of entity specified by the *entity* parameter.	false	No
`servletfiles`	Indicates all servlets should be synchronized regardless of the type of entity specified by the *entity* parameter.	false	No
`webappfiles`	Indicates all Web applications should be synchronized, regardless of the type of entity specified by the *entity* parameter.	false	No
`appfiles`	Indicates all applications should be synchronized regardless of the type of entity specified by the *entity* parameter.	false	No

Option	Description	Default	Required
`clientappfiles`	Indicates all application clients should be synchronized regardless of the type of entity specified by the *entity* parameter.	false	No
`connectorfiles`	Indicates all connectors should be synchronized, regardless of the type of entity specified by the *entity* parameter.	false	No
`syncwebappjavaclasses`	When synchronizing Web application files, whether to include class files included in the Web application's custom class list.	true	No
`newprimary`	Indicates that the sync primary specified with this option overrides the default. This option is required when the sync command is initiated from a different server than the one used previously to connect to the current target.	false	See description
`newversion`	Indicates this sync command should result in a new cluster version number. This option is relevant only if the `cluster` option is also specified.	false	No
`refresh`	Refreshes remote objects after the sync command finishes.	false	No
`restart`	Restarts remote servers after the sync command finishes.	false	No
waittime	Indicates a period of time in seconds to wait for a remote server to restart before restarting the next server. This command is relevant only if the `restart` option is also specified.	0	No
`verbose`	Indicates a verbose server output log.	false	No
clustername	The name of the cluster to be synchronized. Either the `cluster` or the `servers` option must be specified.	-	See description
serverURLs	Server URLs in a comma-delimited list. For example: `serverURL1,serverURL2,serverURL3` Either the `cluster` or the `servers` option must be specified.	-	See description

Option	Description	Default	Required
entity	The name of the entity to be synchronized, in the form *EntityType*:*EntityName* as described in "Entity identifiers" on page 218. EntityType must be one of the following: • Cluster • Server • WebApplication • Application • Connector • Package • Component	-	Yes

Return value

Return value	Indicates
0	The command ran successfully; the result is true/success.
2	The command did not run successfully; an exception was thrown.

Examples

This example synchronizes the estore application on the servers in the cluster MyCluster.

- Command line:

```
jagtool sync -cluster MyCluster Application:estore
```

- Ant build file:

```
<jag_sync cluster="MyCluster" entity="Application:estore" />
```

Usage

If you specify a cluster (using the `cluster` option) when a previous sync command to the cluster came from a different source/primary, then you must specify the `-newprimary` option. Otherwise, the sync command fails.

See also

Chapter 6, "Clusters and Synchronization"

CHAPTER 13

Using Systems Management

This version of Systems Management is based on the Java Management Extensions (JMX) agent management framework. It provides developers with an opportunity to create management beans (MBeans) that can be run in a JMX framework.

Overview

JMX is both an architecture and a set of APIs designed to help manage system components. It allows Java clients to view management-related information, and to perform management operations on these components. For more details about JMX, see the Sun JMX home page at http://java.sun.com/products/JavaManagement.

This version of Systems Management uses the MX4J implementation of the JMX APIs. For information about MX4J, see the MX4J Web page at http://mx4j.sourceforge.net. A JMX system consists of a JMX MBean server and certain services—together known as a JMX agent—and a set of MBeans, which provides the management logic. The MBeans are Java objects that expose management information and operations to clients of the JMX MBean server. These clients can exist within the same VM, or in separate VMs.

The JMX API defines the interfaces that clients must use to access the MBeans. For remote clients, the current JMX release does not fully define the architecture or APIs to be used. In this release, Sybase uses the connector approach provided by the MX4J release. This works over remote method invocations (RMI), and replicates the same VM API, remotely. Remote clients obtain handles on the JMX MBean server, which look like local handles; they implement the same interfaces that local clients can access. Some restrictions exist for remote clients, but the basic JMX API is supported.

Installation directory Throughout this chapter, the Systems Management installation directory is called *ROOT*.

Systems Management functionality

Systems Management includes the following functionality:

- **JMX agent** A JMX agent provides a container that allows MBeans to be managed and accessed remotely, using suitable adapters. It is based on the MX4J agent and complies with JMX API 1.1.
- **SNMP support** This release provides support for the Simple Network Management Protocol (SNMP), which includes an SNMP master agent from J.AgentX.

  The master agent returns information contained in a management information base (MIB). To view an SNMP MIB, you must have an SNMP management console.
- **EAServer agent** This version includes an EAServer MBean and a configuration file, which lists the MBean and its associated initialization.
- **Systems Management Console** The Web-based console allows you to perform systems management.
- **SSL support** You can use SSL when connecting to the JMX agent. The Systems Management Console uses SSL (over RMI) in its interactions.

To use Systems Management, define the JAVA_HOME environment variable to point to a JRE 1.4 installation, and start EAServer using the `-jdk14` option. “Starting the server” on page 49 explains how to use this option.

JMX agent

Once started, the JMX agent consists of a JVM running as a separate process from any other managed application. It acts as a server, having a number of listener threads that are implemented via JMX adapters. These threads include one that responds to SNMP requests and another that responds to RMI requests. The JMX agent contains a number of MBeans, which are accessible via the listeners. The MBeans provide a variety of services, some of which are JMX related. Other services are related to the applications that the JMX agent is being used to manage.

When you start the JMX agent, the Bootstrap MBean is loaded, which loads all the remaining MBeans. The Bootstrap MBean checks the *agent.props* file to determine which MBeans to load and where they are located. See “Creating services” on page 312 for details about setting this up correctly.

The JMX agent in this release is based on the MX4J Release 1.1.1.

The remainder of this section describes the capabilities of the JMX agent in more detail.

Service MBeans

Some of the MBeans deployed into the JMX agent are designated as service MBeans, which all expose a similar lifecycle API, and are all registered with the ServicesManager MBean.

The ServicesManager MBean provides a consistent view of all services and an API for manipulating service MBeans. In addition, the ServicesManager MBean can generate JMX notifications when operations are performed on underlying service MBeans. The ServicesManager MBean monitors the state of each of the underlying MBeans, and if any of their states have changed since the last poll, raises a JMX notification. This means that if one of the service MBeans is a proxy for an external process (for example, EAServer), then the ServicesManager MBean can generate a JMX notification when the remote process fails or stops.

Service proxy MBeans

Some of the underlying MBeans that are registered as services are proxy MBeans for remote services—including one for EAServer and one for SNMP. These MBeans provide a view of remote processes—such as EAServer or the SNMP master agent—and can monitor the state of these remote processes. If a remote process dies, you can obtain the state of the process from the proxy MBean.

You can also use a proxy MBean to start or stop a remote process directly, or to view the process's log file.

Events

The JMX agent includes an MBean that specifically listens for JMX notifications from the ServicesManager and other designated MBeans, and logs them to a file. Users can access this file remotely, via the Systems Management Console, which enables them to view significant events; for example, when a remotely managed server process goes down, or when someone starts a service. Logging events that are generated by other MBeans can also be recorded this way, as well as in the JMX agent log file.

MIB manager MBeans

MIB manager MBeans manage the underlying MIBs. This means that there is one for the NETWORK-SERVICES MIB, and one for the J2EE MIB. The MIB manager MBeans:

- Register the object identifier (OID) sub-tree that is associated with the MIB
- Create the tables defined in the MIB, and populate them with the appropriate rows
- Provide support for rebuilding the tables when required
- Listen for JMX events and map them to SNMP traps

Adapters and connectors

Systems Management includes two adapter MBeans, SNMP and RMI. The SNMP adapter allows you to use an SNMP management console to view SNMP information that is exposed via the deployed MIBs; NETWORK-SERVICES and J2EE. The RMI adapter supports remote access to the JMX agent via RMI. The RMI adapter is used with a connector that replicates the local MBeanServer API to remote clients. See “Connecting to the JMX agent” on page 309 for details about how to write clients that connect to the agent.

JMS forwarding

Systems Management includes a service MBean that can forward JMX notifications as JMS messages.

EAServer proxy MBean

An EAServer proxy MBean is registered as a service MBean, with the ServicesManager MBean, when the JMX agent is started. This allows users to manage services, such as remote EAServer processes, using the ServicesManager MBean. The ServicesManager MBean can use the EAServer proxy MBean to monitor a remote EAServer process for changes in state, and for requests to start or stop the process. The state of registered services, such as the EAServer service (EAS), is available through the ServicesManager MBean, the Systems Management Console, SNMP clients, and JMX clients.

The EAServer proxy MBean provides methods that enable clients to start, stop, restart, and refresh EAServer. In addition, the proxy MBean provides methods that enable clients to retrieve information about EAServer, such as:

- State information (running, stopped, or failed)
- Version information
- Log file details
- Repository information, including component and package details
- Monitor data
- Message service information
- Various other EAServer attributes

SNMP proxy MBean

An SNMP proxy MBean is registered as a service MBean with the ServicesManager MBean when the JMX agent starts, and is subsequently monitored by the same ServicesManager MBean.

Clients of the SNMP proxy MBean can retrieve the SNMP log file, retrieve the state of the SNMP service, and start or stop the service.

SNMP functionality

This release of the Sybase Systems Management framework provides support for SNMP. The Systems Management framework allows SNMP clients (for example, management consoles) to view information about underlying system status using standard SNMP protocols. To access SNMP information in this way, use a management console or an SNMP-enabled client front-end that is capable of interacting with an SNMP master agent. This release includes a master agent that uses the AgentX protocol to interface with the JMX agent that runs on the machine hosting the monitored application. You can replace this master agent with your own, provided it is AgentX-enabled.

SNMP master agent

This version includes the SNMP master agent, which is a public domain SNMP agent, freely obtainable from the J.AgentX Web site at http://eden.dei.uc.pt/agentx. You can also use your own SNMP master agent if it is AgentX-enabled. The SNMP master agent works with the SNMP protocols SNMPv1, SNMPv2, and SNMPv3.

If you use the SNMP master agent that is included in this release, you can specify the ports it uses by supplying suitable arguments to the start-up script *SybMaster.sh* or *SybMaster.bat*, located in *ROOT/bin*. The default SNMP UDP (User Datagram Protocol) port is 7776. The default port used by the AgentX protocol is 7777, and this is the port that the JMX agent connects to using its default configuration file.

❖ **Configuring the SNMP master agent**

If you have an SNMP client tool that listens for traps, you can configure the master agent to send SNMP traps to a designated trap receiver by setting the *SEND_TRAPS* variable:

1. Change to the *ROOT/bin* directory.
2. Open the *master.conf* file and edit the values that you want to change. The following *master.conf* file is included with EAServer:

```
##########################################
# SNMP eXtensible Agent configuration file #
##########################################
VERBOSE=false
SNMP_UDP_PORT=7776
AGENTX_UDP_PORT=7777
#SNMP_UDP_PORT=161
#AGENTX_UDP_PORT=705

# Values to fill in the system node of the MIB
SYS_DESCR=SNMP eXtensible Agent developed at University of Coimbra,
Portugal
SYS_OBJECTID=0.0
SYS_CONTACT=agentx@dei.uc.pt
SYS_NAME=J.AgentX
SYS_LOCATION=UC-PT
SYS_SERVICES=0

# IP and UDP port from the managers that receive the traps
SEND_TRAPS=localhost:162;

# Views over the MIB implemented in the master agent - note that
# the SNMP MIB module is read only.
VIEWS=public:system.read/snmp.read/agentx.read;admin:system.readwrite/
snmp.read/agentx.readwrite;
```

❖ **Starting the SNMP master agent**

- Change to the *ROOT/bin* directory, and enter:

  On UNIX or in an MKS shell on Windows – `SybMaster.sh [-v]`

  On Windows – `SybMaster.bat [-v]`

  Use the `-v` option to generate more verbose output. SybMaster writes output to *master.stdout*, in the EAServer *SysMgmt* directory.

❖ **Stopping the SNMP master agent on UNIX**

- Find the SybMaster process number and kill the process; for example:

  ```
  ps -ef | grep java
  kill -9 <process_ID>
  ```

  You can also run these commands from an MKS shell on Windows.

❖ **Stopping the SNMP master agent on Windows**

- In the SybMaster window, enter Ctrl-C.

SNMP management console

To view SNMP information, you need a management console that connects to the SNMP master agent. The console connects to the SNMP agent on the port you specify when starting the SNMP agent. Various public domain management consoles or client front ends are available, as well as commercially provided ones.

SNMP MIBs

This release includes the following MIBs:

- NETWORK-SERVICES MIB – creates a table with a row for each service that is running in the JMX agent.
- J2EE MIB – contains EAServer-specific tables and columns that expose EAServer statistics.

For each MIB, there is a MibManager MBean, which is responsible for registering the MIB (and its OID sub-tree) with the SNMP master agent, and for managing the creation of tables and scalars that the MIB is composed of.

With the J2EE MIB, EAServer start-up and shutdown events trigger an SNMP trap that is forwarded to the master agent, and then (depending on master agent configuration) to destination trap receivers. This allows management consoles that can receive these traps to be notified quickly when, for example, EAServer has gone down.

NETWORK-SERVICES MIB

The NETWORK-SERVICES MIB was originally defined in RFC (Request For Comments) 1565, and provides a means for listing network-available services through a standard MIB definition.

In this implementation, Sybase populates only the applTable table, creating a row for each service that is registered with the ServicesManager MBean. The service's state is displayed in the applOperStatus column. The service name—as registered with the ServicesManager—is displayed in the applName column. The assocTable table is not used.

J2EE MIB

The J2EE MIB was defined under JSR 77, a Java community program to provide a management framework for application servers. It provides a number of groups, each of which contains a set of related tables, that are used to represent the contents or structure of an application server, and allow SNMP clients to obtain statistical information about the application server.

For this release of the System Management framework, not all the tables in the MIB are populated. Additional EAServer-specific information, not available in the standard J2EE MIB, is provided by extra tables that are included under the J2EE OID sub-tree. The following sections describe the current implementation of the J2EE MIB as shipped with this release.

j2eeMoGroup

The j2eeMoGroup contains tables for various objects and structures located in the application server. For example, it contains information about configured servers, JVMs, and EJBs.

This j2eeMoGroup contains the EAServer-specific tables described in Table 13-1.

Table 13-1: EAServer-specific tables

Table name	Contains information about
j2eeEasNetworkTable	Network connections
j2eeEasMSTable	Message service components
j2eeEasPackageTable	Packages
j2eeEasComponentTable	Components
j2eeEasConnCacheTable	Connection caches
j2eeEasClusterTable	Clusters
j2eeEasConnectedUserTable	Connected IIOP users

j2eeEasStatGroup

The j2eeEasStatGroup contains the EAServer statistics tables described in Table 13-2.

Table 13-2: EAServer statistics tables

Table name	Contains statistics for
j2eeEasNetworkStatTable	Network connections
j2eeEasMSStatTable	Message service entries
j2eeEasMSQueueStatTable	Message queues
j2eeEasMSTopicStatTable	Message topics
j2eeEasMSThreadPoolStatTable	Thread pools
j2eeEasPackageStatTable	Packages
j2eeEasComponentStatTable	Components
j2eeEasConnCacheStatTable	Connection caches

j2eeJvmStatGroup

The j2eeJvmStatGroup includes a single table, j2eeJvmStatTable, which contains statistics for the JVMs that are running.

Unsupported groups

The J2EE MIB groups listed below are not supported in this release:

- j2eeServletStatGroup
- j2eeEjbStatGroup
- j2eeJavaMailStatGroup
- j2eeJtaStatStatGroup
- j2eeJcaStatStatGroup
- j2eeJdbcStatStatGroup
- j2eeJmsStatStatGroup

SNMP MIB implementation details

When a SNMP client, such as an SNMP management console, requests information that is located within one of the supported MIBs, the values that are returned to the client are determined by the MBeans that are deployed in the JMX agent that supports the MIB. The MBeans interact directly with the underlying managed resource to obtain the values. For example, the J2EE MIB provides details associated with EAServer. This information is extracted from EAServer by the EAServer proxy MBean, using EAServer-specific APIs—see “EAServer proxy MBean” on page 301.

Because an SNMP client can generate many requests in rapid succession, it is not reasonable to generate a corresponding request against EAServer for each request that is received from the client. Instead, the JMX agent MBeans cache information retrieved from EAServer, and re-query the server only when the cached data becomes stale. Cached data includes information about the existence of rows in a table (there may be a row for each deployed EJB), and the values in each row. Cached data becomes stale at different rates; by default, 60 seconds for the existence of rows, and 10 seconds for the values in a row. JMX agent MBeans recalculate the existence or rows within a table when 60 seconds have elapsed since the last time the rows were calculated. The MBeans refresh the contents of a row when 10 seconds have elapsed since the last time the row was refreshed. Typically, the structure of a table changes infrequently, if at all; the content of rows is more dynamic.

SNMP MIBs and EAServer 4.*x*

EAServer 4.*x* shipped with an SNMP implementation that included a highly proprietary MIB (SYBASE-Easnew). Most of the information that was provided in this MIB is now located in the EAServer-specific tables of the J2EE MIB described above.

The SYBASE-Easnew MIB enabled you to start and stop EAServer over SNMP, using OIDs. This feature is not supported in the current version of the J2EE MIB. To achieve the same functionality, use the JMX APIs exposed in this release of the Systems Management framework.

Running the JMX agent

Before you run the JMX agent, verify that the JAVA_HOME environment variable points to a JRE 1.4 installation, and that SybSNMP is running so that the agent can connect to it—see “SNMP master agent” on page 302.

❖ **Starting the JMX agent on UNIX**

- Change to the *ROOT/bin* directory, and enter:

  ```
  SybAgent.sh [-F agent props] [-h] [-o log] [-v n]
  ```

  Table 13-3 defines the command line options.

Table 13-3: SybAgent start-up options

Option	Description
-F *path*	Specifies the location of the *agent.props* file, which contains the necesssary configuration to start the agent server. You can make a copy of *agent.props*, then change any of the settings; for example, to define a different class path, or to include other services.
-h	Prints the list of start-up options.
-o *log*	Specifies the name of the log file; the default is *agent.log*.
-v *n*	Writes the debugging messages identified by *n* to the log file identified by -o. Table 13-4 defines the possible values for *n*.

Table 13-4: SybAgent debugging options

Debugging level	Generates these messages
0	FATAL
1	ERROR
2	WARN
3	INFO
4	DEBUG

❖ **Starting the JMX agent on Windows**

- Change to the *ROOT\bin* directory, and enter:

```
SybAgent.bat [-F agent props] [-h] [-o log] [-v n]
```

where Table 13-3 defines the command line options.

❖ **Stopping the JMX agent on UNIX**

- Change to the *ROOT/bin* directory, and enter:

```
StopSybAgent.sh
```

❖ **Stopping the JMX agent on Windows**

- Change to the *ROOT\bin* directory, and enter:

```
StopSybAgent.bat
```

Connecting to the JMX agent

You can access the JMX agent remotely using an RMI connector by writing a client that opens an RMI connection using the classes provided with this release. The client should obtain an object that implements the javax.management.MBeanServer interface over RMI. This interface provides a number of methods that allow the client to interact with the JMX agent, such as invoking methods on MBeans and obtaining MBean information. However, some methods that are exposed in the interface do not work remotely; for example, the registerMBean API call.

This is an example of a client that accesses the JMX agent:

```
package client;


import java.util.Properties;
import javax.management.MBeanServer;
import com.sybase.management.jmx.util.MBeanServerLocator;
import com.sybase.management.jmx.util.MX4JRMIConnectorServerLocator;

// Simple Client to the JMX agent - just prints number of MBeans

public class Client
{
  public static void main(String[] args)
  {
     MBeanServer server = null;

     if (args.length < 4)
     {
        System.err.println("usage: Client <host> <port> <user> <password>");
        System.exit(1);
     }

     String host = args[0];
     String port = args[1];
     String user = args[2];
     String password = args[3];

     Properties props = new Properties();

     props.put(MBeanServerLocator.PROVIDER_HOSTNAME, host);
     props.put(MBeanServerLocator.PROVIDER_PORT, port);
     props.put(MBeanServerLocator.SECURITY_PRINCIPAL, user);
     props.put(MBeanServerLocator.SECURITY_CREDENTIALS, password);
```

```
     // optional: SSL
     //
     // To verify the server's identity, pass in our truststore
     // so that we can check that the server's certificate is valid
     //
     //if (args.length < 5)
     //{
     //    System.err.println("usage: Client <host> <port> <user> <password>
<truststore>");
     //    System.exit(1);
     //}
     //String trustStoreFileName = args[4];
     //props.put(MX4JRMIConnectorServerLocator.TRUST_STORE_FILE,
trustStoreFileName);
     //
     // To verify the client's identity, pass in a pointer to our keystore
     //
     //if (args.length < 6)
     //{
     //    System.err.println("usage: Client <host> <port> <user> <password>
<truststore> <keystore>");
     //    System.exit(1);
     //}
     //String keyStoreFileName = args[6];
     //props.put(MX4JRMIConnectorServerLocator.KEY_STORE_FILE,
keyStoreFileName);

     try
     {
     // get a connection with the remote JMX agent
     server = MX4JRMIConnectorServerLocator.getMBeanServer(props);
     }
     catch (Exception e)
     {
        System.err.println("Failed to connect to the JMX agent: " + e);
        e.printStackTrace(System.err);
        System.exit(1);
     }
     // use the connection ...
     System.out.println("There are " + server.getMBeanCount() + " MBeans");

     System.exit(0);
     }
}
```

Creating services

A service is an object that can run either within or outside of the JMX agent's VM. Systems Management Console users and clients to the JMX agent can control service objects. A service is implemented using a service MBean that is registered with the ServicesManager MBean—see "Service MBeans" on page 299. To create a service:

1 Create a service MBean, which interacts with external services.

2 Create and register a service to represent the MBean.

Creating service MBeans

Service MBeans must expose a well-defined lifecycle API that provides the ability to start, stop, and check the status of managed services. To create a service MBean:

1 In the Systems Management Console, highlight MBeans, right-click, and select Add MBean.

2 Follow the instructions in the Add MBean wizard.

The wizard registers the MBean on the server, and configures the MBean to load on start-up.

Because MBeans are invoked via the JMX MBeanServer APIs, service MBeans are not required to implement a particular interface or extend a particular class to ensure life-cycle compliancy. However, they are required to implement the methods described in Table 13-5.

Table 13-5: MBean required methods

Method	Description
void start()	Starts the MBean.
void stop()	Stops the MBean.
int getState()	Returns the state of the MBean. The return values, as defined in JSR 77, can be: • 0 – STARTING • 1 – RUNNING • 2 – STOPPING • 4 – FAILED

The com.sybase.management.jmx.services.ServiceSupport class is available to extend, if required. This class provides all the API methods described in Table 13-5 and Table 13-6. If you extend this class, you can override the methods defined in Table 13-6, and let the ServiceSupport class handle state management for you. See “Service builder” on page 313 for an example of how to do this. The ServiceSupport class provides default implementations for all the methods listed in Table 13-6, so you can implement only the ones you need. The ServiceSupport class is in *ROOT/lib/sybasejmx.jar*.

Table 13-6: MBean optional methods

Method	Description
void getLogfile()	Gets the service’s log file
void initService()	Initializes the resources
void refreshService()	Refreshes the MBean without restarting
void restartService()	Stops, then restarts the MBean
void startRecursiveService()	Starts dependent MBeans
void startService()	Starts the service
void stopService()	Stops the service
void terminateService()	Cleans up the resources

Creating services

To create and register a service that represents a service MBean:

1 In the Systems Management Console, highlight the Services node in the tree view. Right-click, and select Add Service.

2 Follow the instructions in the Add Service wizard.

The wizard creates the host configuration, connects to the host if set to auto-connect, and writes the configuration to the Systems Management Console’s database (if the user is not anonymous).

Running the samples

The *samples* subdirectory contains samples that illustrate some of the functionality that is provided in this release.

JMX notification listener

The NotificationSample program obtains an RMI connection to the JMX MBeanServer, and listens for JMX notifications generated by the ServicesManager MBean, which are sent whenever a service MBean changes state, or an operation is performed on it. For example, if EAServer is shut down (or crashes), this generates events that are picked up by NotificationSample.

To run the NotificationSample program:

- Change to the *ROOT/bin* directory, and run:

```
runSample.sh samples/NotificationSample <host> <rmiport> <user> <pwd>
```

Where:

- *host* is the host where the JMX agent is running.
- *rmiport* is the port used by the RMI registry; the default is 1099.
- *user* is the RMI user to connect; the default is “sybrmiclient”.
- *pwd* is the password for the user; the default is “_5y6rm1cl1en4”.

EAServer log monitor

The EASLogMonitorSample sample obtains an RMI connection to the JMX MBeanServer, and monitors the contents of the EAServer error log, checking for a string that, if seen, triggers a service restart, using the service interface exposed over JMX. To run the program:

- Change to the *ROOT/bin* directory, and run:

```
runSample.sh samples/EASLogMonitorSample [-v] [-rebootString <string>] \
<host> <rmiport> <user> <pwd>
```

Where:

- `-v` indicates that the contents of the log file are to be printed.
- `-rebootString <`***string***`>` sets the reboot string, which can trigger a service restart; the default is “NO_PERMISSION”.
- *host* is the host where the JMX agent is running.
- *rmiport* is the port used by the RMI registry; the default is 1099.
- *user* is the RMI user to connect; the default is “sybrmiclient”.
- *pwd* is the password for the user; the default is “_5y6rm1cl1en4”.

Service builder

SimpleMBean extends com.sybase.management.jmx.services.ServiceSupport, a Sybase-provided abstract class, which you can use to build services. To use SimpleMBean, you must first register it with the JMX MBeanServer. You can either:

- Use an XML file to load it when the JMX MBeanServer starts, or

- Dynamically install it later, using the JMX MLET service. To implement this approach, read the appropriate JMX documentation.

To implement the first approach:

1 Create a JAR file called *simple.jar* that contains the *Simple.class* and *SimpleMBean.class* files, which are created when you compile the sources. Put *simple.jar* in the *lib* subdirectory.

2 Create an XML file called *test.xml*, similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>

<mbeans>

  <mbean name="test:Name=Test"
    code="samples.Simple"
    archive="file:XXX-ROOT-XXX/lib/simple.jar">
  </mbean>

  <service name="Test"
    type="test"
    mbean="test:Name=Test"
    startmode="manual">
  </service>

</mbeans>
```

Replace *XXX-ROOT-XXX* with the location of the Systems Management installation.

3 Edit the *ROOT/agent.props* file, and set the value of the products property to:

```
com.sybase.management.jmx.boot.products=
(name=EAS,url=file:XXX-ROOT-XXX/eas.xml),
(name=NetSNMP,url=file:XXX-ROOT-XXX/netsnmp.xml),
(name=Test,url=file:XXX-ROOT-XXX/test.xml)
```

Replace *XXX-ROOT-XXX* with the location of the system management installation. The products property now refers to the *test.xml* file that you just created.

4 Stop and restart the JMX MBeanServer.

The new Test service should be visible through the Systems Management Console, and you should be able to use the console to start and stop the service, and verify that its state changes appropriately.

Compiling the samples

To build all the samples, which also puts the class files in the *classes* subdirectory, change to the *ROOT/bin* directory, and run:

On UNIX – `compileSamples.sh`

On Windows – `compileSamples.bat`

Using SSL

Using SSL allows clients to establish secure links to a JMX agent. Using SSL ensures that the client-server link is secure, and that the client has indeed set up a link to the correct server. However, it does not authenticate the client; the server does not know who the client is.

Further authentication—for example, identifying that the client is who it purports to be—must be performed separately. Only clients that have a copy of the server's signed certificate can connect, so you can restrict connections to the set of clients to whom the certificates have been granted.

Setting up SSL

To set up SSL, you must generate a keystore to be used by the JMX agent's RMI adapter. For detailed information about keystores, see the Sun Web page at http://java.sun.com/products/jdk/1.2/docs/api/java/security/KeyStore.html. Use this syntax to generate the keystore for each JMX agent installed in the network:

```
keytool [-alias <alias>] \
-genkey -v -keystore <keystore> \
-storepass <storepwd> -keypass <keypass> \
-dname <dname> \
-validity <validity>
```

Where:

- The keytool executable is located in the *jdk/jdk1.4/bin* subdirectory of your EAServer installation.
- <*alias*> is the alias used for the key. If omitted, it defaults to "mykey".
- <*keystore*> identifies the keystore.

- <*storepwd*> is the password used to protect the keystore itself.
- <*keypass*> is the password used to protect the key being generated.
- <*dname*> is a distinguished name; for example:

  ```
  "CN=Chris Jobson, O=Sybase\, Inc., C=UK"
  ```

- <*validity*> is the number of days the key is valid; for example, 365 (one year).

This example generates a keystore and a key, then puts the new key in the keystore:

```
keytool -genkey -v -keystore key.store -storepass storepwd -keypass keypwd \
-dname "CN=Chris Jobson, O=Sybase\, Inc., C=UK" -validity 365
```

Next, you must export the X.509 certificate that authenticates the key you just created:

keytool [-alias <*alias*>] -export \
-v -storepass <*storepwd*> -keystore <*keystore*>
-file <*certfile*>

Where, in addition to the variables described in the previous example:

- <*certfile*> is the name of the certificate file.

This example exports the X.509 certificate:

```
keytool -export -v -storepass storepwd -keystore -file x509cert
```

Once you have exported the key (in the X.509 certificate), you must import it into the truststore used by the clients that connect to the RMI adapter. Currently, the only clients run in servlets that are deployed in EAServer. Import the certificate into a truststore located in the servlet's *WEB-INF/lib* directory, located under *$JAGUAR/Repository/WebApplication/SysMgmtm/*. You must copy the *x509cert* file to the same machine where the servlet is deployed, and then import it using this syntax:

keytool [-alias <*alias*>] -import \
-v -storepass <*storepwd*> -keystore <*truststore*>
-file <*certfile*>

Where, in addition to the previously defined variables:

- <*truststore*> is the truststore to be used by the clients.

This example creates a truststore, which contains a single certificate that wraps the original key, and puts the X.509 certificate in *C:\tmp\x509cert*:

```
keytool -import -v -storepass storepwd -keystore trust.store \
```

```
-file C:\tmp\x509cert
```

Note Delete the X.509 certificate after using it.

Do this for all clients that will connect to the JMX agent. After creating the various keys and truststores, you must ensure that the JMX agent starts the RMI adapter so that it uses SSL. To do this, make these changes to the *boot.xml* file:

1 Locate the definition for the sybase.system.adaptor.service:Name=SSLServerSocketFactory MBean, then set the appropriate values for its KeyStoreName, KeyStorePassword, and KeyManagerPassword attributes:

2 Locate the definition for the sybase.system.adaptor:Protocol=RMI-JRMP MBean, and to ensure that the JMX agent starts and the RMI adapter accepts only SSL connections, verify that the SSLFactory attribute is set to "sybase.system.adaptor.service:Name=SSLServerSocketFactory".

The RMI adapter exports its certificate, which is checked by the client using its truststore. Therefore, the client can verify that the server is who it purports to be.

To configure the servlet to use SSL to connect:

1 Change to the *ROOT/WEB-INF/classes/com/sybase/servlet/jmx* directory.

2 In each <*agent*>*.properties* file—there is one for each agent that is defined—verify that the com.sybase.management.jmx.adaptor.rmi.truststore.resource property is set to point to the truststore that contains the trusted certificate for the appropriate JMX agent. For example:

```
com.sybase.management.jmx.adaptor.rmi.truststore.resource=
/clienttrust.store
```

`/clienttrust.store` is loaded by appropriate class loaders, and should load the truststore created by the keytool -import command. To load a truststore from another location, verify that it can be loaded by the class loader used to load the servlet, or use one of these alternate properties, and set the location appropriately:

- com.sybase.management.jmx.adaptor.rmi.truststore.url – requires a property value in this format:

  <*protocol*>:<*value*>

  For example, file:<*location*> or http://<*host*>:<*port*>/*location*.

- com.sybase.management.jmx.adaptor.rmi.truststore.file – the property value must be a path to the truststore.

Each JMX agent requires its own trusted certificate; they can all be stored in the same truststore, or in separate ones.

Note If you are not using SSL, remove the trust_store property from the properties file.

The server can also verify whether the client is presenting a valid certificate. This is the inverse of the process described above. The server has a truststore it consults to verify that the client's certificate matches. To set this up, create keystores for the clients, and matching truststores for the server (containing the exported key). Follow the steps outlined above using keytool, then:

1 In *boot.xml*, locate the definition for the com.sybase.system.adaptor.service:Name=SSLServerSocketFactory MBean, then edit the values for the TrustStoreName, TrustStorePassword and NeedClientAuth attributes appropriately:

2 On the client end, edit the *agent.properties* file, and set the value of the com.sybase.management.jmx.adaptor.rmi.keystore.resource property to point to the keystore containing the certificate for the relevant JMX agent. The resource is loaded by the appropriate class loaders, and should load the keystore that was created using the keytool -genkey command.

   Alternately, set the values for these properties:

   - com.sybase.management.jmx.adaptor.rmi.keystore.url – requires a property value in this format:

     <*protocol*>:<*value*>

     For example, file:<*location*> or http://<*host*>:<*port*>/*location*.

   - com.sybase.management.jmx.adaptor.rmi.keystore.file – a path to the keystore.

   - com.sybase.management.jmx.adaptor.rmi.keymanager.password – the key manager password used to create the key that is located in the keystore.

Using the Systems Management Console

The Systems Management Console allows you to monitor events across a multimachine architecture or in a standalone environment, and to view the status of individual services and agents. With the appropriate permissions, you can also use the Systems Management Console to perform administrative tasks, such as starting or stopping a service.

Connecting to the Systems Management Console

To start, stop, or restart services, the Systems Management Console interacts with the JMX agent running in each installation. The agent is itself a lightweight server that accepts network requests from the Systems Management Console.

Note Since Systems Management is an EAServer application, you cannot use the Systems Management Console to start or restart the server that is hosting the Systems Management application.

❖ **Connecting to the Systems Management Console**

1 In your Web browser, go to:

```
http://host:8080/WebConsole
```

Where *host* is the name of the server that hosts the Systems Management application. The default host is the machine name.

2 In the login window, enter:

- User Name – a valid EAServer user name.
- Password – the password for the user name.

The user interface controls

Once you are logged in, you see a multi-pane window that consists of:

- Banner – contains links that allow the user to log out, and to obtain online help.
- Footer – displays the user who is currently logged in.

- Details view (right pane) – displays detailed information about a selected item in the tree view, and allows you to perform operations on services.
- Tree view (left pane) – contains a hierarchical overview of the various components that are accessible through the console. There is a node under JMX Browser for each host server that the console knows about. Under each of these nodes are the following folders:
  - **Services** The Services folder contains all the services that have been installed. A service is simply an MBean with a well-defined lifecycle that has been registered with the ServicesManager MBean. All services have the same interface presented through the Systems Management Console. For each service, you can view basic information about it, and perform a number of operations on it, such as starting it, or viewing its log file.
  - **Events** The Events folder allows you to view events that have been recorded on the JMX agent. These are events that correspond to JMX notifications, raised by certain system components.
  - **MBeans** The MBeans folder provides access to the underlying MBeans, including those MBeans that are displayed as services. For information about the MBeans whose attributes you can configure and whose operations you can perform, see “Accessing MBeans” on page 325.

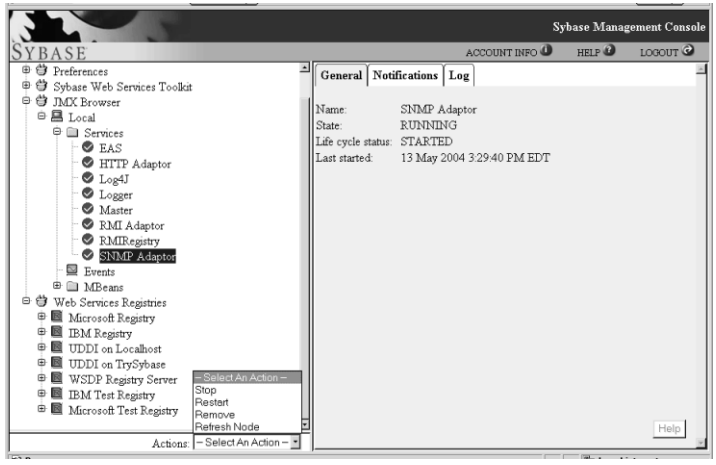
  You can highlight a node in the tree view, and see details associated with it in the property view.

When you are logged in to the Systems Management Console, the MBeans and services that are visible are determined by the J2EE roles that you have been granted. A service can be associated with certain read and update roles. This means that users must be granted these roles before they can view, or perform operations on, the service. For example, the EAServer service might have a read role called ReadX, as well as the default Admin Role. To view the EAServer service, users must be granted either the ReadX role or the Admin Role. By default, a service can be operated on only by someone who has the Admin Role—see “Managing user roles” on page 324.

Navigating the tree view

When you first log in, the tree view is collapsed. Use the plus sign to expand the tree view; your Web browser displays two panes similar to those shown in Figure 13-1. The tree view in the left pane expands and collapses to show the hierarchical view of managed services. When you highlight a specific item in the tree, detailed information about the item displays in the details view in the right pane. In the tree view, you can expand items by clicking on the plus-sign icons next to each folder, and collapse the hierarchy by clicking on the minus-sign icons. Items that do not display a plus-sign icon cannot be expanded.

Figure 13-1: Systems Management Console



The details view

The details view displays information about the item you highlight in the tree view.

JMX Browser

When you select JMX Browser in the tree view, the details view displays a list of the installed servers.

Servers

When you select a server in the tree view, the details view displays the following information:

- Profile Name – the name of the server profile, such as “Local.”
- Host Name – the name of the machine hosting the server.
- Port – the port where the server listens for connections; the default is 1099.
- Principal – user name for the JMX agent on the remote machine.

- Credentials – password for the JMX agent on the remote machine.
- Trust Store – name of the trust store.
- Key Store – name of the key store file.
- Password – password for the key store and trust store.
- Connect at login – check box; by default, unchecked.

Services

When you select a service in the tree view, the details view displays these tabs:

- **General** Information about the service.
  - Name – the name of the service, such as “SNMP Adaptor.”
  - State – the state of the service, which can be:

    STARTING
    RUNNING
    STOPPING
    STOPPED
    FAILED

  - Life Cycle Status – the current state of the service’s lifecycle, which can be:

    NOTINSTALLED
    INSTALLED
    INITIALISED
    STARTED
    STOPPED
    TERMINATED
    FAILED

  - Last Started – the time the service was started.
- **Notifications** This section displays notifications generated in connection with the service; for example, if the SNMP master agent goes down. If this happens, State displays as “FAILED” but the Life Cycle Status displays as “STARTED,” which means that the service started but has since gone down. Life Cycle Status indicates what has been done to the service; for example, started or stopped. State displays the current state of the service. If a service fails, a warning triangle also displays next to the service name in the tree view.
- **Log** Displays the log. To display only part of the log, enter the number of lines you want to see.

Events

When you select Events in the tree view, the details view displays a list of the most recent events. To display a subset of the events, enter the number of lines you want to see.

MBeans

When you select MBeans in the tree view, the details view displays a list of the installed MBeans. When you select a specific MBean in the tree view, the details view displays the MBean attributes, which vary according to the MBean. “Accessing MBeans” on page 325 describes the MBeans you can configure and the operations you can perform.

Performing actions

To perform an action on one of the entities in the tree view, highlight the entity, click Actions at the bottom of the page, and select the appropriate action from the list. Table 13-7 defines the actions for each entity type.

Table 13-7: Entity actions

Entity	Actions
JMX Browser	• Add Host (server) • Refresh Node
A specific server	• Disconnect • Remove • Refresh Node
Services	• Add Service • Refresh Node
A specific service	• Start • Stop • Restart • Refresh Node • Remove
Events	• Reset • Refresh Node
MBeans	• Add MBean • Refresh Node
A specific MBean	• Refresh Node

Managing user roles

User roles are determined by the EAServer security system, including Enterprise Security, if it is installed. Roles are not part of the Systems Management framework.

By default, users who have been granted the Admin Role can view all the services and MBeans that are deployed in the JMX agent, and can perform operations, such as starting and stopping services. You can also configure your system to allow users with roles other than the Admin Role to perform these tasks.

Managing access to services

Tighter controls exist for the roles required to access or operate on a service when it is accessed via the Services folder in the Systems Management Console. To configure these roles, you must edit the XML file that is used to start and configure the service. Every service can be assigned a set of read roles, and another set of update (execute) roles. If a service has no associated roles, it defaults to the values defined on the servlet's Init-Params tab.

To assign a set of roles to a service, edit the appropriate XML file, and change the service definition. The XML files are located in the Systems Management *ROOT* directory; the EAServer service XML file is *eas.xml*.

Assume that you want to require that to view the EAServer service, users must have been granted either the ReadRole or Admin Role, and to do anything to the service (execute a method, or change an attribute) users need either the UpdateRole or Admin Role. To configure this scenario, the contents of *eas.xml* should look something like this:

```
<service name="EAS"
   type="product"
   mbean="sybase.system.service:Type=EAS,Name=Jaguar"
   startmode="automatic"
   readroles="ReadRole, Admin Role"
   updateroles="UpdateRole, Admin Role"
</service>
```

Instead of specifying "Admin Role," you can use "+" to indicate the default role or roles specified on the servlet's Init-Params tab.

In this example, to see and update the EAServer service, you must be granted either Admin Role, or both ReadRole and UpdateRole. If you are granted only ReadRole, you can view the EAServer service details but not update it. If you are granted only UpdateRole, you cannot see the service, so you cannot update it either.

For more information about user roles, see the *EAServer Security Administration and Programming Guide*. If Enterprise Security is installed, see also the *Enterprise Security Administration Guide*.

Accessing MBeans

Using the Web Console, you can access the following MBeans to manage your system:

- sybase.system:Name=Master
- sybase.system.jms:Type=Forwarder
- sybase.system.adaptor.security:Name=UserPassword
- sybase.system.logger:Type=File
- sybase.system.log4j:Type=Notifier
- sybase.system.service:Name=Jaguar,Type=EAS

The attributes you can modify and the operations you can perform for each of these MBeans is described below.

sybase.system:Name=Master

Description

The sybase.system:Name=Master=Master MBean controls the JMX Agent process.

Attribute name	Description	Datatype
LogPath	Path and name of the log file	String
InstallDir	EAServer installation directory	String

stop

Description	Shuts down the JMX Agent process.
Syntax	stop()
Return value	void

sybase.system.jms:Type=Forwarder

Description

The sybase.system.jms:Type=Forwarder MBean forwards notifications that occur within the MBeanServer to a configurable location in the JMS domain (either a topic or queue). You can specify one or more MBeans to be monitored, using the addMonitoredMBean method.

This MBean supports the EAServer implementation of JMS only. An initial context connection is made to iiop://<*JMS_host*>:<*JMS_port*>, then the com.sybase.jms.InitialContextFactory is used. For more information, see Chapter 31, "Using the Message Service," in the *EAServer Programmer's Guide*.

Attribute name	Description	Datatype
JMSHost	Host name of the machine running the JMS server	String
JMSPort	Connection port number on the machine running the JMS server	String
JMSTopic	The name of the topic where notification messages are published	String

addMonitoredMBean

Description	Add an MBean that you want to monitor.
Syntax	addMonitoredMBean(*mbean_object* javax.management.ObjectName)
Parameters	*mbean_object* Name of the MBean object that you want to monitor.
Return value	void

sybase.system.adaptor.security:Name=UserPassword

Description The sybase.system.adaptor.security:Name=UserPassword MBean is used to set up the remote method invocation (RMI) authentication credentials, which are used by the RMI adapter.

Attribute name	Description	Datatype
User	A user's login name	String
Password	The user's password	String
RMIAdaptor	The name of the RMI adaptor	String

sybase.system.logger:Type=File

Description The sybase.system.logger:Type=File MBean listens for JMX notifications, converts them to log messages, and writes them to a log file. You can also use this MBean to read and truncate the log file.

Attribute name	Description	Datatype
User	A user's login name	String
Password	The user's password	String
RMIAdaptor	The name of the RMI adaptor	String

addMonitoredMBean

Description Adds an MBean for which to receive notifications.

Syntax addMonitoredMBean(*mbean_object* javax.management.ObjectName)

Parameters *mbean_object*
Name of the MBean object for which to receive notifications.

Return value void

retrieveLog

Description	Retrieves the specified number of lines from the log file.
Syntax	retrieveLog(*lines* int)
Parameters	*lines* The number of lines to retrieve from the log file.
Return value	java.lang.String[]

truncateLog

Description	Removes all but the specified number of lines from the log file.
Syntax	truncateLog(*lines* int)
Parameters	*lines* The number of lines to leave in the log file.
Return value	void

sybase.system.log4j:Type=Notifier

Description

The sybase.system.log4j:Type=Notifier MBean creates a new Log4J appender, to which Log4J messages can be written. This MBean detects Log4J messages and converts them to JMX notifications. This makes it possible to send a Log4J message that was created anywhere in the JVM, as a JMX notification. To set this up, create a suitable entry in the *log4j.properties* file, and specify that debug messages created in the required hierarchy are sent to the JMXAppender.

Reviewers: What is the syntax for an entry in log4j.properties?

Attribute name	Description	Datatype
MinLoggingLevel	The minimum level at which to map Log4J messages to JMX notifications	int

start

Description	Starts the Log4J to JMX notification mapper.
Syntax	start(void)
Return value	void

stop

Description	Stops the Log4J to JMX notification mapper.
Syntax	stop(void)
Return value	void

sybase.system.service:Name=Jaguar,Type=EAS

Description: The sybase.system.service:Name=Jaguar,Type=EAS MBean is a client of the EAServer *Monitoring* CORBA component. All the statistics provided by the *Monitoring* component are accessible via this MBean.

Attribute name	Description	Default value	Datatype
Host	Server host machine.	Name of the machine hosting the server	String
InstallDir	The EAServer installation directory.		String
JDK	The JDK version with which EAServer is running.		String
LogPath	The full path and name of the server's log file.	*$JAGUAR/bin/Jaguar.log*	String
Password	User's login password.		String
Port	IIOP connection port number.	9000	int
ReconnectDelay	The minimum delay required between successive attempts at reconnecting to EAServer if the first attempt fails.	5000	long
ServerName	The name of the server.	Jaguar	String
StartCommand	Command that was used to start the server.		String

Attribute name	Description	Default value	Datatype
StartTimeout	If EAServer takes longer than this specified number of seconds to start, it is assumed that EAServer failed to start.	60000	long
User	Login user's name.	jagadmin	String

init

Description	Initializes the sybase.system.service:Name=Jaguar,Type=EAS MBean.
Syntax	start(void)
Return value	void

listEntityChildren

Description	Returns the data from the entity's items() method.
Syntax	listEntityChildren(*entityType* java.lang.String, *entityName* java.lang.String)
Parameters	*entityType* The entity type. *entityName* The entity name.
Return value	java.lang.String[][]

listEntityProperties

Description	Lists the contents of the properties file for the specified entity type and entity name.
Syntax	listEntityProperties(*entityType* java.lang.String, *entityName* java.lang.String)
Parameters	*entityType* The entity type. *entityName* The entity name.
Return value	java.util.Properties

listMSNames

Description	Lists the entries for the specified Message Service component.
Syntax	listMSNames(*ms_component* java.lang.String)
Parameters	*ms_component* Must be one of: • ActiveQueues • ActiveTopics • ConfiguredQueues • ConfiguredTopics • ThreadPools
Return value	java.lang.String[]

listNetworkTypeNames

Description	Returns the listener types that you can configure with this MBean.
Syntax	listNetworkTypeNames(void)
Return value	java.lang.String[IIOP, IIOPS, HTTP, HTTPS, TDS, TDSS]

refresh

Description	Refreshes the MBean.
Syntax	refresh(void)
Return value	void

restart

Description	Restarts the MBean.
Syntax	restart(void)
Return value	void

retrieveComponentMonitorData

Description	Retrieves monitoring data about the specified EAServer component.
Syntax	retrieveComponentMonitorData(*componentName* java.lang.String)
Parameters	*componentName* The name of the component for which to retrieve monitoring data.
Return value	double[]

retrieveConnCacheMonitorData

Description	Retrieves monitoring data about the specified connection cache.
Syntax	retrieveConnCacheMonitorData(*cacheName* java.lang.String)
Parameters	*cacheName* The name of the connection cache for which to retrieve monitoring data.
Return value	double[]

retrieveConnectedUsers

Description	Retrieves information about connected users from the EAServer *Monitor* component.
Syntax	retrieveConnectedUsers(void)
Return value	com.sybase.jaguar.system.UserInfo[]

retrieveLog

Description	Retrieves the specified number of lines from the log file, beginning with the line number identified by the first parameter.
Syntax	retrieveLog(*line_number* int, *total_lines* int)
Parameters	*line_number* The line number of the first line to retrieve. *total_lines* The total number of lines to retrieve.
Return value	java.lang.String[]

retrieveLog

Description	Retrieves the specified number of lines from the log file, beginning with the first line.
Syntax	retrieveLog(*lines* int)
Parameters	*lines* The number of lines to retrieve from the log file.
Return value	java.lang.String[]

retrieveMSData

Description	Retrieves the Message Service runtime monitoring data.
Syntax	retrieveMSData(*ms_component* java.lang.String, *instanceName* java.lang.String)
Parameters	*ms_component* The Message Service component type, which can be one of: • MessageQueue • MessageTopic • ThreadPool *instanceName* The name of the Message Service component instance.
Return value	double[]

retrieveNetworkData

Description	Retrieves runtime monitoring data for the specified listener type.
Syntax	retrieveNetworkData(*listenerType* java.lang.String)

Parameters	*listenerType* The listener type, which can be one of: • IIOP • IIOPS • HTTP • HTTPS • TDS • TDSS
Return value	double[]

retrievePackageMonitorData

Description	Retrieves monitoring data about the specified EAServer package.
Syntax	retrievePackageMonitorData(*packageName* java.lang.String)
Parameters	*packageName* The name of the package for which to retrieve monitoring data.
Return value	double[]

start

Description	Starts an MBean instance.
Syntax	start(void)
Return value	void

startRecursive

Description	Starts all MBean instances.
Syntax	startRecursive(void)
Return value	void

stop

Description	Stops an MBean instance.
Syntax	stop(void)
Return value	void

terminate

Description	Terminates all MBean instances.
Syntax	terminate(void)
Return value	void

APPENDIX A

Using EJB 1.0 JAR Support

This appendix describes EAServer Manager's support for EJB 1.0 JAR files. EAServer supports EJB 1.0 for backward compatibility. The EJB 2.0 model is recommended for new development.

Importing and exporting EJB 1.0 JAR files

❖ Importing an EJB 1.0 JAR file

This feature is provided for backward compatibility with previous EAServer versions and any EJB 1.0 servers. To import the JAR file:

1. Start EAServer Manager if it is not already running, and connect to the server where you want to install the component.
2. Highlight the top-level Packages folder. Choose File | Deploy | EJB 1.0 JAR.
3. Enter the path to the JAR file, and optionally enter a package name. If you do not specify a package name, EAServer installs the components into a package with the same name as the base JAR file name. For example, components imported from *EmploymentAuth.jar* are installed to package *EmploymentAuth*.
4. EAServer Manager displays the beans that are defined in the JAR file. Select and configure beans for deployment as follows:
   a. In the left column, select the check boxes for each bean that you want to deploy, or click Select All to deploy all beans.
   b. Highlight each selected bean, then click Configure. Verify the deployment descriptor properties described in "EJB 1.0 deployment descriptor properties" on page 338.

5 When all selected beans have been configured, click Deploy.

6 Map role names that were read from the deployment descriptor to role names that exist in EAServer Manager. Names in the left column are used in the deployment descriptor; these may not match existing roles in EAServer Manager, or may match roles that do not agree with the access control requirements intended for the bean. For each name in the left column, assign a role name using the drop-down list in the right column.

7 Map run-as identity names that were read from the deployment descriptor to identity names that exist in EAServer Manager. You can configure these mappings now or later. To configure now, for each name in the left column, assign an identity name using the drop-down list in the right column. To configure after deployment completes, use the Run-As Mode tab in the Component Properties or Method Properties dialog.

8 Optionally generate stubs and skeletons for the bean. You must generate stubs and skeletons before the bean can run, but you can do so after deployment completes. If generating now, specify a code base for the generated files. Sybase recommends the EAServer *java/classes* subdirectory, which is the default.

❖ **Exporting an EJB 1.0 JAR file**

1 Highlight the EAServer package to export and choose File | Export, then choose EJB 1.0 JAR.

2 Enter the path and file name for the new JAR file and click Next.

3 EAServer Manager creates the JAR file, displaying status messages in the Export wizard.

EJB 1.0 deployment descriptor properties

When importing beans from an EJB-JAR file, EAServer Manager displays the settings from each bean's deployment descriptor and allows you to make changes before the information is recorded in EAServer's configuration repository.

The information is displayed in the Verify Enterprise Bean Settings dialog box, which contains the tabs listed below. Changes that you make in the Verify Enterprise Bean Settings dialog box do not affect the settings recorded in the EJB-JAR file, only the settings to be recorded in EAServer's configuration repository.

After deploying an EJB component into EAServer, you can change these properties using the controls in the Component Properties dialog box.

BeanName tab

The leftmost tab defines the general settings for the bean. The bean's implementation is dependent on the bean type and the Java class names specified here. You may change the EJB Home Name if you wish. Do not change other settings unless the deployment descriptor was prepared incorrectly.

- **EJB Class** The Java class that implements the bean, specified in dot notation.
- **Home Interface Class** The Java home interface name, specified in dot notation.
- **Remote Interface Class** The Java remote interface name, specified in dot notation.
- **EJB Home Name** The name suffix used by client applications to look up the bean's home interface in the EAServer naming service. The full name consists of the server's initial naming context followed by a slash (/) and the bean's home name.
- **EJB Session Type** *Session beans only.* Whether the session bean is stateful or stateless.
- **EJB Primary Key** *Entity beans only.* The Java class that specifies primary key values for the entity bean, in Java dot notation.
- **Reentrant** *Entity beans only.* Whether the bean is **reentrant**. A reentrant bean can participate in loopback call sequences, which are call sequences where one of the bean's methods calls another component which in turn calls a method in the calling bean instance. Most beans are not implemented to support reentrancy. Do not enable this option unless the bean developer has verified that the implementation allows it.

Changing general properties after deployment After deployment completes, use the General tab in the Component Properties dialog to view or modify these properties. The Reentrant property for an entity bean is specified with a check box on the Instances tab.

Access Control tab

This tab configures the users that can call the bean's methods. If roles are listed for a method name, only users in that role can execute the method. The displayed role names are those listed in the bean's deployment descriptor. The displayed roles may not exist in EAServer Manager. If they do not, the Deployment Wizard prompts you to map these role names to existing EAServer Manager roles after all selected beans have been configured.

Changing Access Control properties after deployment completes After deployment completes, you can configure access control for the EJB component and its methods in EAServer Manager. To change access control for the component, use the Role Membership folder below the component's icon. To change access control for a method, expand the Interfaces folder below the component icon, then expand the icon for the interface that contains the method. Add or remove roles using the Role Membership folder.

Control Descriptor tab

Use this tab to configures settings that control how EAServer executes the bean's methods:

- **Tx Attribute** Determines how the bean participates in transactions managed by EAServer. For more information, see the online help or Chapter 4, "Defining Components," in the *EAServer Programmer's Guide*.
- **Tx Isolation** If the bean participates in transactions, determines the transaction isolation level. For more information, see the online help or Chapter 4, "Defining Components," in the *EAServer Programmer's Guide*.

- **Run-as Mode** Determines the user name and password used when the bean calls other EAServer components installed in the same server or server cluster. The choices are:

Option	Description
Client	Use the client's username and password.
System	Use the system user name and password. The system account effectively belongs to any role, and can execute any method on any component that is installed in the same server or cluster.
Specified	Use the user name and password associated with the identity name specified in the Run-as Identity column.

- **Run-as Identity** If the run-as mode is set to Specified, Run As Identity must specify an identity name to authenticate calls to components that are installed on the same server or server cluster. The identity name read from a bean may not exist in EAServer. If it does not, the Deployment wizard asks you to map this identity name to an identity listed in the EAServer Manager Identities folder after all selected beans have been configured.

Changing Control Descriptor properties after deployment completes
After deployment completes, you can configure transactional properties on the Transactions tab in the Component Properties dialog box. Use the Run-As Mode tab to view and edit the run-as mode and run-as identity settings.

Environment tab

Use this tab to specify bean properties required by the implementation. For example, there might be a property specifying a JDBC database URL for queries to a remote database. At runtime, the bean accesses these properties using the EJBContext.getEnvironment() method. The tab initially displays the properties and values defined in the bean's deployment descriptor. You can modify, add, and delete properties as follows:

To do this	Do this
Modify a property	Edit the value in the right column.
Add a property	Click Add. A blank row appears. Type a property name and a value for the property. Property names beginning with com.sybase are reserved for Sybase internal use.
Delete a property	Highlight the property name or value column, then click Delete.

Changing Environment properties after deployment completes After deployment completes, you can configure environment properties using the Advanced tab in the Component Properties dialog box. The environment properties are listed in addition to the com.sybase.jaguar.component properties used internally by EAServer.

APPENDIX B

Repository Properties Reference

About these reference pages

These reference pages document EAServer properties as they are stored in the configuration repository. Typically, if you configure EAServer using EAServer Manager, you do not need to work with properties in this format. You will edit properties in this format if:

- You use jagtool or jagant to configure EAServer, as described in Chapter 12, “Using jagtool and jagant.”
- You use an EAServer XML configuration file to configure properties inside a J2EE archive file, as described in “Using EAServer configuration files in J2EE archives” on page 196.
- You use the Advanced tab in EAServer Manager to configure properties that cannot be set any other way.
- You write applications that use the Jaguar::Management and Jaguar::Repository interfaces to programmatically configure properties. For information on these interfaces, see the generated HTML Interface Repository documentation, located in the *html/ir* subdirectory of your EAServer installation.

User-defined properties

You can define additional properties that are not listed here. For example, if you are implementing CORBA components, you can define additional properties that are read by the component implementation.

Encrypted properties

If a property name ends in `.e`, values for the property are encrypted when stored in the EAServer configuration repository. You can use this feature to encrypt user-defined properties. You cannot rename properties that are used internally by EAServer to use encryption, because EAServer will continue to look up the property with the original name.

"See also" headings

Some properties include a section called "See also," which directs you to a location where you can find more information. Cross-references to other properties within this appendix do not include a page number, because the properties are listed alphabetically. Cross-references to other chapters in this book include either the chapter number or the page number.

Application properties

Description — Application property names are prefixed with `com.sybase.jaguar.application`. You can configure application properties in EAServer Manager using the Application Properties Dialog box and the folders displayed under the application icon.

com.sybase.jaguar.application.applicationclients

Description — Specifies the comma-separated list of J2EE application clients that are installed in this application.

Syntax — *appclient1*, *appclient2*, ...

Where *appclient1*, *appclient2*, and so forth are names of J2EE application clients.

Usage — In EAServer Manager, set this property by selecting the application's Application Clients folder and using the File menu.

com.sybase.jaguar.application.classloaderpolicy

Description — Specifies how the custom class loader (version 2) resolves version conflicts when you specify the same class at multiple levels in the class loader hierarchy.

Syntax — One of the following values:

- `ParentFirst` (the default) to delegate to the parent class loader before attempting to load the class at the level of this entity.
- `ParentLast` to attempt to load the class at the level of this entity before delegating to the parent.

The value is ignored unless using version 2 of the custom class loader; that is, com.sybase.jaguar.server.jvm.classloader must be set to `JaguarClassLoaderv2`.

See also — com.sybase.jaguar.server.jvm.classloader, com.sybase.jaguar.component.classloaderpolicy, com.sybase.jaguar.package.classloaderpolicy, com.sybase.jaguar.webapplication.classloaderpolicy

Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.application.description

Description — Contains a text description of the application.

Syntax — *desc*

Where *desc* is the descriptive text.

Usage — In EAServer Manager, set this property using the Description field on the General tab of the Application Properties dialog box.

com.sybase.jaguar.application.DOMfactory

Description — Specifies the class name for a custom DOM XML parser factory class.

Syntax — The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.application.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.

See also: com.sybase.jaguar.application.SAXfactory, com.sybase.jaguar.application.XSLTfactory

com.sybase.jaguar.application.java.classes

Description: Specifies Java classes and JAR files to be loaded by the application's custom class loader.

Syntax: A comma-separated list of Java classes, packages, and JAR files. You can specify all classes in a package using wildcards, as in this example:

```
com.xyz.MyPackage.*
```

You can specify all classes in a JAR file by specifying the JAR file name, as in this example:

```
MyEntityBean.jar
```

The JAR file must be deployed in the EAServer *java/classes* subdirectory.

Usage: See "Custom class lists for packages, applications, or servers" in Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*.

In EAServer Manager, set this property using the Java Classes tab in the Application Properties dialog box.

See also: com.sybase.jaguar.package.java.classes, com.sybase.jaguar.component.java.classes, com.sybase.jaguar.server.classloader.debug, com.sybase.jaguar.server.java.classes, com.sybase.jaguar.webapplication.java.classes

com.sybase.jaguar.application.large-icon

Description: Specifies the name of an icon file to represent the Web application. This property is not used in EAServer, but is preserved in applications that are imported from a J2EE EAR file.

Syntax: *icon-file*

Where *icon-file* is the name of the icon file.

Usage: In EAServer Manager, use the Advanced tab to set this property.

com.sybase.jaguar.application.name

Description	Specifies the application name.
Syntax	*app-name* Where *app-name* is the application name.
Usage	In EAServer Manager, the application name is set when creating or importing an application and cannot be changed.

com.sybase.jaguar.application.packages

Description	Specifies the comma-separated list of packages that are installed in this application.
Syntax	*package1*, *package2*, ... Where *package1*, *package2*, and so forth are names of packages.
Usage	In EAServer Manager, set this property by selecting the Application's Packages folder and using the File menu.

com.sybase.jaguar.application.SAXfactory

Description	Specifies the class name for a custom SAX XML parser factory class.
Syntax	The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.application.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.
See also	com.sybase.jaguar.application.DOMfactory, com.sybase.jaguar.application.XSLTfactory

com.sybase.jaguar.application.security.identities

Description	Specifies the list of trusted identities used for incoming component invocations when propagating client credentials from another server.
Syntax	*t1*,*t2*,*t3*,... Where *t1*, *t2*, and so forth are identity names that are defined in the repository.

See also
com.sybase.jaguar.application.security.identity,
com.sybase.jaguar.server.security.identities,
Security properties on page 500

com.sybase.jaguar.application.security.identity

Description
Specifies the identity used for outgoing component invocations when propagating client credentials to another server.

Syntax
The name of an identity that is defined in the repository.

See also
com.sybase.jaguar.application.security.identities,
com.sybase.jaguar.server.security.identity,
Security properties on page 500

com.sybase.jaguar.application.security-role.<j2ee-role>

Description
Specifies a mapping from a J2EE role name used in the application to a role defined in the EAServer repository.

Syntax
com.sybase.jaguar.application.security-role.*j2ee-role=jag-role*

Where:

- *j2ee-role* is the role name used in the application and listed in the com.sybase.jaguar.application.security-roles property.
- *jag-role* is the role name defined in the EAServer repository.

Usage
In EAServer Manager, set this property using the Role Mapping tab in the Application Properties dialog box.

Role names can also be specified at the package, component, or Web application level.

See also
com.sybase.jaguar.application.security-roles,
com.sybase.jaguar.package.security-role.<j2ee-role>,
com.sybase.jaguar.package.security-roles,
com.sybase.jaguar.webapplication.security-role.<j2ee-role>,
com.sybase.jaguar.webapplication.security-roles,
com.sybase.jaguar.component.security-role-ref.<j2ee-role-ref>,
com.sybase.jaguar.component.security-role-refs

com.sybase.jaguar.application.security-roles

Description — Specifies logical J2EE role names used in the application.

Syntax — *role1*, *role2*, ...

Where *role1*, *role2*, and so forth are of the form:

(description=*role-desc*, name=*role-name*)

Where *role-desc* is an optional description of the role, and *role-name* is the name used in the application.

Usage — In EAServer Manager, set this property using the Role Mapping tab in the Application Properties dialog box. Each J2EE role name must be mapped to an EAServer role name by setting or creating the corresponding com.sybase.jaguar.application.security-role.<j2ee-role> property.

J2EE role names can also be specified at the package, component, or Web application level.

See also — com.sybase.jaguar.application.security-role.<j2ee-role>, com.sybase.jaguar.webapplication.security-roles, com.sybase.jaguar.package.security-roles com.sybase.jaguar.package.security-role.<j2ee-role>, com.sybase.jaguar.component.security-role-ref.<j2ee-role-ref>, com.sybase.jaguar.component.security-role-refs

com.sybase.jaguar.application.small-icon

Description — Specifies the name of an icon file to represent the Web application. This property is not used in EAServer, but is preserved in applications that are imported from a J2EE EAR file.

Syntax — *icon-file*

Where *icon-file* is the name of the icon file.

Usage — In EAServer Manager, use the Advanced tab to set this property.

com.sybase.jaguar.application.webapplications

Description — Specifies the comma-separated list of Web applications that are installed in this application.

Syntax — *webapp1*, *webapp2*, ...

Where *webapp1*, *webapp2*, and so forth are names of Web applications.

Usage — In EAServer Manager, set this property by adding or removing items from the application's Web Applications folder.

com.sybase.jaguar.application.XSLTfactory

Description — Specifies the class name for a custom XSLT XML parser factory class.

Syntax — The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.application.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.

See also — com.sybase.jaguar.application.DOMfactory, com.sybase.jaguar.application.SAXfactory

Application client properties

Description — Application client property names begin with `com.sybase.jaguar.applicationclient`. In EAServer Manager, you can configure application client properties in the Application Client Properties dialog box.

com.sybase.jaguar.applicationclient.description

Description — Contains a text description of the application client.

Syntax — *desc*

Where *desc* is the descriptive text.

Usage — In EAServer Manager, set this property using the Description field on the General tab in the Application Client Properties dialog box.

com.sybase.jaguar.applicationclient.ejb-ref

Description — Specifies a list of EJB references that define aliased JNDI names for EJB components invoked by the application client. Application clients must use JNDI to resolve EJB references, using the prefix `java:comp/env/ejb` in JNDI lookups.

Syntax — *ejb-ref1*, *ejb-ref2*, ...

Where *ejb-ref1*, *ejb-ref2*, and so forth follow the syntax of EJB reference properties on page 447.

Usage — In EAServer Manager, configure this property using the EJB References tab in the Application Client Properties dialog box.

com.sybase.jaguar.applicationclient.env-entry

Description — Environment properties allow you to specify global read-only data for use by the application client. Application clients must use JNDI to retrieve environment properties, using the prefix `java:comp/env` in JNDI lookups.

Syntax — See Environment properties on page 452.

Usage — In EAServer Manager, set this property using the Environment tab in the Application Client Properties dialog box.

com.sybase.jaguar.applicationclient.files

Description — Specifies additional files and Java classes to be distributed with the application client.

Syntax — A list of file names, separated by commas. Files may be specified as follows:

- Specify Java classes and packages using the Java dot notation. For example, `com.sybase.CORBA` adds all files in the com.sybase.CORBA package. These classes must be deployed under the EAServer *java/classes* subdirectory.
- If a DLL or shared library is deployed in the EAServer *cpplib* subdirectory, you can enter the file name itself. For example, `myutils.dll`.
- Other files must be specified using full paths, paths that are relative to the EAServer *Repository* subdirectory, or paths qualified by environment variables. For example:
  - *../dll/debug/MyDebugLibrary.dll*

- *d:\mydir\myfile.ext*
- *{$JAGUAR}/java/src/com/yours/**

If you use full paths, you can only synchronize or import package archives on machines that share the same directory structure as your development machine.

If you use environment variables, use the syntax shown in the example. Environment variables must be set on all machines where you import the exported JAR file, or to which you synchronize.

Usage — In EAServer Manager, set this property using the Files tab in the Application Client Properties dialog box.

com.sybase.jaguar.applicationclient.main-class-name

Description — Specifies the main Java class of the application client.

Syntax — *class*

Where *class* is the main Java class of the application client, specified in dot notation; for example, `com.sybase.appclient.Myclient`.

Usage — In EAServer Manager, set this property using the Main Class Name field on the General tab in the Application Client Properties dialog box.

com.sybase.jaguar.applicationclient.name

Description — Specifies the application client name.

Syntax — *app-name*/*client-name*

Where:

- *app-name* is the name of the J2EE application to which the application belongs.
- *client-name* is the name of the application client.

com.sybase.jaguar.applicationclient.resource-env-ref

Description — Resource environment references are logical names applied to objects administered by EAServer.

Syntax	See Resource environment reference properties on page 489.
Usage	In EAServer Manager, set this property using the Resource Env Refs tab in the Application Client Properties dialog box.
See also	Resource environment reference properties on page 489.

com.sybase.jaguar.applicationclient.resource-ref

Description	Specifies aliased JNDI names for database connections, JavaMail sessions, and URL factories used by the application client.
Syntax	See Resource reference properties on page 496.
Usage	In EAServer Manager, set this property using the Resource Refs tab in the Application Client Properties dialog box.
See also	Resource reference properties on page 496.

Cluster properties

Description	Cluster property names begin with `com.sybase.jaguar.cluster`. A cluster configures load balancing and failover support for an application that is deployed to two or more host machines. In EAServer Manager, configure cluster properties in the Cluster Properties dialog box.
See also	Chapter 6, “Clusters and Synchronization”

com.sybase.jaguar.cluster.bindpassword

Description	Specifies the bind password for cluster name servers.
Syntax	*password* Where *password* is the bind password.
Usage	When you create a cluster, a random bind password is automatically generated. In most cases, you do not need to edit the password. However, if a name server is used by two or more clusters, you must configure the clusters to use the same bind password.

In EAServer Manager, set this property using the Advanced tab in the Cluster Properties dialog box.

com.sybase.jaguar.cluster.DLBbroadcastInterval

Description	When using adaptive load balancing, specifies how often each server broadcasts its load metrics.
Syntax	The broadcast interval, in minutes. The default is 5.
See also	com.sybase.jaguar.cluster.DLBpolicy

com.sybase.jaguar.cluster.DLBcalculateInterval

Description	When using adaptive load balancing, specifies how often the load collectors in the name servers calculate and generate a normalized load list (NLL) of all member servers.
Syntax	The interval, in minutes. The default is 10.
See also	com.sybase.jaguar.cluster.DLBpolicy

com.sybase.jaguar.cluster.DLBenable

Description	Enables configuration of the dynamic load balancing policies (weighted or adaptive).
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.cluster.DLBpolicy

com.sybase.jaguar.cluster.DLBmaxWeight

Description	The maximum weight, used in a weighted load balancing policy, of any server.
Syntax	An integer value. The default is five. The maximum value is ten.
See also	com.sybase.jaguar.cluster.DLBpolicy, com.sybase.jaguar.cluster.DLBweights

com.sybase.jaguar.cluster.DLBpolicy

Description

Specifies the algorithm to distribute component invocations between the servers in the cluster.

Syntax

One of the following values:

Value	To indicate
Random (the default)	Random load distribution. Clients are assigned to servers by picking the server at random. Load distribution should average to equal values over time.
Round-Robin	Round-robin load distribution. Clients are assigned to servers in sequence.
Weighted	Random load distribution, with server weights assigned by the com.sybase.jaguar.cluster.DLBweights property. Server loads should average to proportions of the assigned weights.
Adaptive	Adaptive load distribution, based on load metrics that are configured by these properties: • com.sybase.jaguar.cluster.DLBbroadcastInterval • com.sybase.jaguar.cluster.DLBcalculateInterval • com.sybase.jaguar.cluster.DLBrefreshInterval • com.sybase.jaguar.cluster.DLBsampleInterval

You must set the com.sybase.jaguar.cluster.DLBenable property to true to use value Weighted or Adaptive.

See also

com.sybase.jaguar.cluster.DLBenable,
com.sybase.jaguar.cluster.DLBbroadcastInterval,
com.sybase.jaguar.cluster.DLBcalculateInterval,
com.sybase.jaguar.cluster.DLBmaxWeight,
com.sybase.jaguar.cluster.DLBrefreshInterval,
com.sybase.jaguar.cluster.DLBsampleInterval,
com.sybase.jaguar.cluster.DLBweights

com.sybase.jaguar.cluster.DLBrefreshInterval

Description

When using adaptive load balancing, specifies how often each name server obtains the NLL from its local load collector. All name servers have their own copies of the NLL.

Syntax	The interval, in minutes. The default is 10. The refresh interval must be equal to or greater than the calculate interval specified by com.sybase.jaguar.cluster.DLBcalculateInterval.
See also	com.sybase.jaguar.cluster.DLBpolicy, com.sybase.jaguar.cluster.DLBcalculateInterval

com.sybase.jaguar.cluster.DLBsampleInterval

Description	When using adaptive load balancing, specifies how often each server collects its load metrics.
Syntax	The sampling interval, in seconds. The default is 5.
See also	com.sybase.jaguar.cluster.DLBpolicy

com.sybase.jaguar.cluster.DLBweights

Description	When using the weighted load balancing policy, specifies the relative weights assigned to each server.
Syntax	A comma separated list of server URLs and integer weight values, in the form: *url1*|*weight1*,*url2*|*weight2*, ... Where: • *url1*, *url2*, and so forth, are the IIOP URLs for the servers in the cluster. • *weight1*, *weight2*, and so forth, are the integer weights assigned to each server in the cluster. No value can exceed the value of com.sybase.jaguar.cluster.DLBmaxWeight.
See also	com.sybase.jaguar.cluster.DLBenable, com.sybase.jaguar.cluster.DLBmaxWeight, com.sybase.jaguar.cluster.DLBpolicy

com.sybase.jaguar.cluster.initialcontext

Description	Specifies the initial naming context for the cluster naming services.
Syntax	*context* Where *context* is the initial naming context, for example, "/UK/BusinessComponents".

Usage — In EAServer Manager, set this property using the Advanced tab in the Cluster Properties dialog box.

com.sybase.jaguar.cluster.ipmirrorgroups

Description — Specifies mirror groups used for in-memory component state replication.

Syntax — Specify a comma-separated list of mirror pairs, each of the form:

url1|*url2*

Where *ur1l* and *url2* are the IIOP listener URLs for the servers in the mirror pair, using host names. For example, this value specifies two mirror pairs:

```
iiop://mypc:9000|iiop://yourpc:9100,iiop://hispc:9500|iiop://herpc:9600
```

Each server in a mirror pair must be a member of the cluster, and one server cannot be a member of more than one mirror pair. Servers in a mirror pair should have the same set of stateful components installed.

Usage — In EAServer Manager, configure this property on the Mirror Pairs tab in the Cluster Properties dialog box.

See also — com.sybase.jaguar.cluster.ipmirrorgroups

com.sybase.jaguar.cluster.mirrorgroups

Description — Specifies mirror groups used for in-memory component state replication.

Syntax — Same as for com.sybase.jaguar.cluster.ipmirrorgroups, except that you must use IP addresses rather than host names.

See also — com.sybase.jaguar.cluster.ipmirrorgroups

com.sybase.jaguar.cluster.name

Description — Specifies the cluster name.

Syntax — *name*

Where *name* is the cluster name.

Usage — In EAServer Manager, this property is set when the cluster is created, and cannot be changed.

com.sybase.jaguar.cluster.nameservers

Description — Specifies the cluster's name servers as a list of IIOP URLs.

Syntax — *ns1*, *ns2*, ...

Where *ns1*, *ns2*, and so forth are IIOP URLs of the form:

iiop://*host*:*port*

Where *host* is the host name and *port* is the IIOP port number. For example:

```
iiop://bigbox:9000
```

Usage — In EAServer Manager, set this property using the Name Servers tab in the Cluster Properties dialog box.

See also — "Adding a name server to a cluster" on page 126, "Removing a server from a cluster" on page 127

com.sybase.jaguar.cluster.primary

Description — Specifies the IIOP URL of the primary cluster, which is the source server of the last synchronize operation.

Syntax — iiop://*host*:*port*

Where *host* is the host name and *port* is the IIOP port number.

Usage — This property is set automatically when you synchronize a cluster, and should be considered read-only.

In EAServer Manager, the primary server is displayed in the Primary Server field on the General tab in the Cluster Properties dialog box.

See also — "Synchronization" on page 131

com.sybase.jaguar.cluster.servers

Description — Specifies the servers in the cluster as a list of IIOP URLs.

Syntax — *url1*, *url2*, ...

Where *url1*, *url2*, and so forth are IIOP URLs of the form:

iiop://*host*:*port*

Where *host* is the host name and *port* is the IIOP port number. For example:

```
iiop://bigbox:9000
```

Usage	In EAServer Manager, set this property using the Servers tab in the Cluster Properties dialog box.
See also	“Configuring a server to enable synchronization” on page 124, “Adding a server to a cluster” on page 125, “Removing a server from a cluster” on page 127

com.sybase.jaguar.cluster.startup

Description	Specifies what form of cluster version control member servers perform before joining a cluster.
Syntax	*startup_check* Where *startup_check* is one of: • `check_primary` (the default) to specify that the server version must be equal to the primary server’s version. • `check_servers` to specify that the server version must be equal to a majority of servers in the cluster. • `disable_check` to disable cluster version checking (not recommended).
Usage	In EAServer Manager, set this property on the Advanced tab in the Cluster Properties dialog box.
See also	com.sybase.jaguar.cluster.version “Cluster start-up options” on page 130

com.sybase.jaguar.cluster.updated

Description	Contains audit information for previous synchronize operation.
Syntax	*audit_text* Where *audit_text* is text describing the last synchronize operation.
Usage	This property is set when you synchronize a cluster and should be considered read-only. In EAServer Manager, view the property using the Advanced tab in the Cluster Properties dialog box.
See also	“Synchronization” on page 131

com.sybase.jaguar.cluster.version

Description	Specifies the version number of the cluster. The version number automatically increments each time you synchronize the cluster.
Syntax	*version* Where *version* is the version number. The default for a new server is 0.
Usage	This property is set when you synchronize a cluster and should be considered read-only. In EAServer Manager, view the property using the Advanced tab in the Cluster Properties dialog box.
See also	"Synchronization" on page 131, "Cluster start-up options" on page 130, com.sybase.jaguar.cluster.startup

Component properties

Description	Component property names begin with `com.sybase.jaguar.component`. In EAServer Manager, configure component properties in the Component Properties dialog box.

com.sybase.jaguar.component.auto.failover

Description	Specifies whether client proxies can transparently fail over to a different server.
Syntax	Allowable values are `true` and `false`.
Usage	In EAServer Manager, set this property using the "Automatic failover" check box on the Transactions tab.
See also	Chapter 7, "Load Balancing, Failover, and Component Availability"

com.sybase.jaguar.component.auto.profiles

Description	Obsolete.
Syntax	Not applicable.
Usage	While required to support automatic failover in some earlier EAServer versions, this property is no longer used.

See also com.sybase.jaguar.component.auto.failover

com.sybase.jaguar.component.bind.naming

Description Specifies the name with which the component is bound to the name service. If not specified, the default is *package*/*component* where *package* is the EAServer package name, and *component* is the component name.

Syntax *name*

Where *name* is the name service path, relative to the server or cluster's initial naming context.

Usage In EAServer Manager, you can set this property in the Component Properties dialog as follows:

- For EJB components, enter the value in the bean Home Name field on the General tab.
- For components of other types, use the Advanced tab.

See also com.sybase.jaguar.server.CosNaming.initialcontext, com.sybase.jaguar.cluster.initialcontext

com.sybase.jaguar.component.bind.object

Description Specifies whether instances are bound to client's object reference.

Syntax `true` or `false`. The default is `false`.

Usage This property cannot be enabled unless the component is stateful and thread-safe.

In EAServer Manager, set this property using the Bind Object check box on the Instances tab in the Component Properties dialog box.

See also com.sybase.jaguar.component.thread.safe, com.sybase.jaguar.component.stateless, com.sybase.jaguar.component.tx_vote

com.sybase.jaguar.component.bind.thread

Description Specifies whether instances are bound to the thread that created them.

Syntax `true` or `false`. The default is `false`.

Usage

Set this property to true if your component uses thread-local storage. This property must be enabled for ActiveX components.

In EAServer Manager, set this property using the Bind Thread check box on the Instances tab in the Component Properties dialog box.

com.sybase.jaguar.component.classloaderpolicy

Description

Specifies how the custom class loader (version 2) resolves version conflicts when you specify the same class at multiple levels in the class loader hierarchy.

Syntax

Same as com.sybase.jaguar.application.classloaderpolicy.

See also

com.sybase.jaguar.server.jvm.classloader,
com.sybase.jaguar.application.classloaderpolicy,
com.sybase.jaguar.package.classloaderpolicy,
com.sybase.jaguar.webapplication.classloaderpolicy

Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.component.cmp_iso_level

Description

Specifies the effective transaction isolation level for EJB CMP entity beans.

Syntax

See "Configuring CMP isolation level" in the *EAServer Performance and Tuning Guide*.

See also

com.sybase.jaguar.server.cmp_iso_level

com.sybase.jaguar.component.cmp.version

Description

For EJB 2.0 entity beans that use CMP (automatic persistence), specifies the CMP model version.

Syntax

Allowable values are:

Value	To indicate
1.1 (the default)	CMP according to the EJB 1.1 specification.
2.0	CMP according to the EJB 2.0 specification.

If no value is specified, the default is 1.1.

com.sybase.jaguar.component.code.set

Description	For PowerBuilder, C, or C++ components, specifies the coded character set name used to encode character and string parameter data.
Syntax	*codeset* Where *codeset* is the coded character set name. The default is the package character set specified by com.sybase.jaguar.package.code.set. For the list of the supported values, list the subdirectories of the *charsets* directory. Each subdirectory matches the name of a supported character set.
Usage	In EAServer Manager, set this property using the Code set field on the General tab in the Component Properties dialog box.
See also	com.sybase.jaguar.package.code.set

com.sybase.jaguar.component.com.progid

Description	Specifies the progid that the component uses in the COM Automation Server Registry.
Syntax	*progid* Where *progid* is the COM progid.
Usage	In EAServer Manager, set this property using the Prog ID field on the General tab in the Component Properties dialog box.

com.sybase.jaguar.component.context

Description	Specifies the name of the component's context IDL interface: for example, CtsComponents::ObjectContext.
Syntax	*context* Where *context* is the name of the context IDL interface.
Usage	The context interface must match the context interface used in the component's control interface. For example, if the control interface is CtsComponents::ObjectControl, specify CtsComponents::ObjectContext.

In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.

See also com.sybase.jaguar.component.control

com.sybase.jaguar.component.control

Description Specifies the name of the component's control IDL interface, for example CtsComponents::ObjectControl.

Syntax *control*

Where *control* is the name of the control IDL interface.

Usage The control interface defines methods called by EAServer in response to changes in the instance life cycle. The choices are summarized in this table:

Control interface	Description
JaguarEJB::EntityBean	For EJB entity beans.
JaguarEJB::StatefulSessionBean	For EJB stateful session beans.
JaguarEJB::StatelessSessionBean	For EJB stateless session beans.
JaguarEJB::ServerBean	The EAServer 1.1 Java component life cycle model. The default for Java/CORBA components that do not have persistent state (that is, when the Persistence field is None).
CtsComponents::ObjectControl	A CORBA life cycle model based on the EJB entity bean model. The default for Java/CORBA and C++/CORBA components with persistent state (that is, when the Persistence field is Component Managed).
JaguarCOM::ObjectControl	For ActiveX components.

These interfaces are documented in the generated IDL documentation, which is available in HTML format in the *html/ir* subdirectory of your EAServer installation. If you use a control interface other than JaguarEJB::ServerBean, EAServer generates the control interface methods in the implementation template when you generate a C++ or Java skeleton.

In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box. For EJB components, the control interface is set correctly when you specify the component type property in EAServer Manager, and should not be changed.

See also	com.sybase.jaguar.component.context, com.sybase.jaguar.component.pooling, com.sybase.jaguar.component.stateless, com.sybase.jaguar.component.tx_vote

com.sybase.jaguar.component.cpp.class

Description	For a C++ component, specifies the implementation class name.
Syntax	*class* Where *class* is the name of the implementation class.
Usage	In EAServer Manager, set this property in the C++ Class field on the General tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.cpp.library

com.sybase.jaguar.component.cpp.copy

Description	For C and C++ components, specifies whether the server should copy the component library before running it.
Syntax	`true` or `false`. The default is `false`.
Usage	Set this property to true to allow updates to the implementation on operating systems that do not allow overwriting a DLL or shared library while the library is in use. In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.refresh

com.sybase.jaguar.component.cpp.debug

Description	For a C++ component, specifies whether to catch exceptions.
Syntax	`true` or `false`. The default is `true`, which specifies that exceptions are caught in the server.

Usage

When debugging an executing component, set this property to false to allow exceptions to reach your debugger. You must set this property to true to debug an executing C++ component in Microsoft Visual C++. Other C++ debuggers may require the same setting as well.

In a production server, set the property to true so exceptions thrown by component code do not terminate the server process.

In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.

com.sybase.jaguar.component.cpp.library

Description

For C and C++ components, specifies the name of the DLL or shared library that contains the implementation class.

Syntax

library

Where *library* is the library name, minus platform extensions such as *.dll* for Windows or *.so* for Solaris. You can append `${JAGUAR_PLATFORM}` to the name. `${JAGUAR_PLATFORM}` indicates that this part of the name should be replaced with the EAServer platform identifier (specified by the `JAGUAR_PLATFORM` variable set in the generated makefile). This substitution allows you to deploy libraries for multiple platforms to support multiplatform clusters.

The library must be located in the EAServer *cpplib* directory to support component refresh.

Usage

In EAServer Manager, set this property in the DLL Name field on the General tab in the Component Properties dialog box.

See also

com.sybase.jaguar.component.cpp.class

com.sybase.jaguar.component.cpp.process

Description

For C++ components, specifies the name of an external process to run the component. If the property is not set, the component executes within the EAServer process.

Syntax

executable

Where *executable* is the name of the executable file, minus platform extensions (such as *.exe* for Windows). The executable must be located in the EAServer *cpplib* subdirectory.

In EAServer Manager, set this property in the C++ Executable field on the General tab in the Component Properties dialog box.

com.sybase.jaguar.component.cs.create

Description	For a C component, specifies whether the implementation has a create routine.
Syntax	`true` or `false`. The default is `false`.
Usage	The Jaguar server calls create when creating a new instance of the component. The signature for create is: `CS_RETCODE CS_PUBLIC create()` create must return CS_SUCCEED. In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.cs.destroy

com.sybase.jaguar.component.cs.destroy

Description	For a C component, specifies whether the implementation has a destroy routine.
Syntax	`true` or `false`. The default is `false`.
Usage	The Jaguar server calls destroy when destroying an instance of the component. The signature for destroy is: `CS_RETCODE CS_PUBLIC destroy()` destroy must return CS_SUCCEED. In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.cs.create

com.sybase.jaguar.component.db.sequence

Description	For an EJB CMP entity bean, or components of other types that use automatic persistence, specifies the database sequence name, if required by the database.

Syntax — The syntax used depends on the database you are using. See "Enabling automatic key generation" in Chapter 27, "Creating Entity Components," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.component.debug

Description — Specifies whether the server logs trace information for instance life cycle events such as creation, destruction, pooling, and so forth.

Syntax — `true` or `false`. The default is `false`.

In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.cpp.debug, com.sybase.jaguar.component.pb.debug, com.sybase.jaguar.component.trace

com.sybase.jaguar.component.defer

Description — In an EJB CMP entity bean that uses a Sybase CMP wrapper driver, specifies which SQL commands may be deferred to the end of the transaction.

Syntax — Specify "none" or a comma-separated list of these commands: `insert`, `delete`, `update`. When using a Sybase CMP wrapper JDBC driver, the default is `insert,delete,update` to specify that insert, delete, and update commands are all deferred to the end of the transaction. When not using a Sybase CMP wrapper driver, the default is "none".

Usage — If inserts are deferred, create methods do not throw DuplicateKeyException. Instead the insert fails when the transaction is committed. This behavior is allowed by the EJB 2.0 specification, but may be incompatible with applications coded for EJB 1.1. To use the EJB 1.1 behavior, remove `insert` from the list of deferrable commands.

See also — "Using CMP JDBC wrapper drivers" in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.component.destroyPooledInstancesOnShutdown

Description — Overrides the com.sybase.jaguar.server.destroyPooledInstancesOnShutdown server property.

Syntax	If the server property com.sybase.jaguar.server.destroyPooledInstancesOnShutdown is `true`, you can set this component property to `false` to disable destruction of pooled instances of this component. The component property is ignored if the server property is not set to true.
See also	com.sybase.jaguar.component.destroyPooledInstancesOnShutdownTimeout, com.sybase.jaguar.server.destroyPooledInstancesOnShutdown

com.sybase.jaguar.component.destroyPooledInstancesOnShutdownTimeout

Description	How long to wait for each instance destruction method to return when destroying pooled instances during server shutdown.
Syntax	The time to wait, in seconds. If no value is specified, the default is the value of the server property com.sybase.jaguar.server.destroyPooledInstancesOnShutdownTimeout.
See also	com.sybase.jaguar.component.destroyPooledInstancesOnShutdown, com.sybase.jaguar.server.destroyPooledInstancesOnShutdown, com.sybase.jaguar.server.destroyPooledInstancesOnShutdownTimeout

com.sybase.jaguar.component.dn.triggers

Description	For EJB CMP entity beans that use instance and query caching with database change notification enabled, this property enables automatic creation of database triggers to notify the EAServer cache manager when the table data changes.
Syntax	`true` or `false`. The default is `false`.
Usage	In EAServer Manager, set this property using the Create Database Triggers check box on the Persistence/General tab in the Component Properties dialog box.
See also	Chapter 27, “Creating Entity Components,” in the *EAServer Programmer’s Guide*.

com.sybase.jaguar.component.DOMfactory

Description	Specifies the class name for a custom DOM XML parser factory class.

Syntax	The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.component.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.
See also	com.sybase.jaguar.application.SAXfactory, com.sybase.jaguar.application.XSLTfactory

com.sybase.jaguar.component.ejb.home

Description	For EJB components, specifies the home interface class name. Derived from com.sybase.jaguar.component.home.
Syntax	*class* Where *class* is the Java class name, for example, com.sybase.jaguar.sample.GlossaryHome.
Usage	This property is read-only. It is automatically set when you specify the component's IDL home interface with the com.sybase.jaguar.component.home property. In EAServer Manager, view this property in the Home Interface field on the General tab in the Component Properties window.
See also	com.sybase.jaguar.component.home, com.sybase.jaguar.component.ejb.remote

com.sybase.jaguar.component.ejb.key

Description	For an EJB entity bean component, specifies the class name for the primary key type. Derived from com.sybase.jaguar.component.key.
Syntax	*class* Where *class* is the Java class name; for example, com.sybase.jaguar.sample.GlossaryKey or java.lang.String.
Usage	This property is read-only. It is automatically set when you specify the component's IDL primary key type with the com.sybase.jaguar.component.key property. In EAServer Manager, view this property in the Primary Key field on the General tab in the Component Properties window.
See also	com.sybase.jaguar.component.key

com.sybase.jaguar.component.ejb.local

Description	For EJB 2.0 components, specifies the Java local interface name.
Syntax	*class* Where *class* is the Java class name; for example, com.sybase.jaguar.sample.LocalGlossary.
Usage	This property is read-only. It is automatically set when you specify the component's IDL remote interface with the com.sybase.jaguar.component.local property. In EAServer Manager, view this property in the Local Interface field on the General tab in the Component Properties window.
See also	com.sybase.jaguar.component.local, com.sybase.jaguar.component.ejb.local.home, com.sybase.jaguar.component.ejb.remote

com.sybase.jaguar.component.ejb.local.home

Description	For EJB 2.0 components, specifies the Java local home interface name.
Syntax	*class* Where *class* is the Java class name, for example com.sybase.jaguar.sample.LocalGlossaryHome.
Usage	This property is read-only. It is automatically set when you specify the component's IDL local home interface with the com.sybase.jaguar.component.local.home property. In EAServer Manager, view this property in the Local Home Interface field on the General tab in the Component Properties window.
See also	com.sybase.jaguar.component.local, com.sybase.jaguar.component.ejb.home

com.sybase.jaguar.component.ejb.remote

Description	Specifies the remote interface class name for an EJB component. Derived from com.sybase.jaguar.component.remote.
Syntax	*class*

Where *class* is the Java class name; for example com.sybase.jaguar.sample.Glossary.

Usage

This property is read-only. It is automatically set when you specify the component's IDL remote interface with the com.sybase.jaguar.component.remote property.

In EAServer Manager, view this property in the Remote Interface field on the General tab in the Component Properties window.

See also

com.sybase.jaguar.component.remote, com.sybase.jaguar.component.ejb.home

com.sybase.jaguar.component.ejb-local-ref

Description

For EJB components, specifies a list of EJB local references that define aliased JNDI names for local components invoked by this component.

Syntax

ejb-ref1, *ejb-ref2*, ...

Where *ejb-ref1*, *ejb-ref2*, and so forth follow the syntax of EJB local reference properties on page 447.

Usage

Use this property if aliased beans are invoked using local interfaces. If you are using remote interfaces, use the com.sybase.jaguar.component.ejb-ref property.

See also

EJB local reference properties on page 447, com.sybase.jaguar.component.ejb-ref

com.sybase.jaguar.component.ejb-ref

Description

For EBJ components, specifies a list of EJB references that define aliased JNDI names for components invoked by this component.

Syntax

ejb-ref1, *ejb-ref2*, ...

Where *ejb-ref1*, *ejb-ref2*, and so forth follow the syntax of EJB reference properties.

Usage

Use this property if aliased beans are invoked using remote interfaces. If you are using local interfaces, use the com.sybase.jaguar.component.ejb-local-ref property.

See also

EJB reference properties on page 447

com.sybase.jaguar.component.env-entry

Description	For EJB components (EJB version 1.1 or later), specifies environment properties.
Syntax	See Environment properties on page 452.
Usage	In EAServer Manager, configure environment properties on the Environment tab in the Component Properties dialog box. EJB 1.0 components do not use this property. Any property whose name does not begin with com.sybase.jaguar.component is considered an environment property for an EJB 1.0 component.

com.sybase.jaguar.component.external.request.timeout

Description	For components that run in an external server, specifies how long to wait for a response from the external server before returning an error to the client.
Syntax	The timeout in seconds. If not set, the default is the value of the server property com.sybase.jaguar.server.external.serverstart.timeout for the server where the component is installed. A value of 0 specifies infinity.
See also	com.sybase.jaguar.component.external.servername, com.sybase.jaguar.server.external.request.timeout

com.sybase.jaguar.component.external.servername

Description	For stateless components, specifies the server name for external component execution.
Syntax	The server name. The external server must run on the same machine as the server to which your application's clients connect. If not set, the component executes in the same process as the server in which it is installed. This property is ignored if the component is not stateless (com.sybase.jaguar.component.stateless is false).

Usage — Running a component externally protects the server process from application problems such as memory leaks or segmentation violations. Stateless components of any type can run externally, with full access to server-side features such as cached connections. When you mark a component to run externally, EAServer runs it in a separate server process. You can identify which external server runs the component, and assign groups of related components to run in the same external server. EAServer starts the external server when required, and restarts the server if it stops responding.

See also — com.sybase.jaguar.component.external.request.timeout, com.sybase.jaguar.component.external.serverstart.timeout, com.sybase.jaguar.component.stateless

com.sybase.jaguar.component.external.serverstart.timeout

Description — If the component is configured to run in an external server, specifies how long to wait for a response from the external server before returning an error to the client.

Syntax — The timeout in seconds. If not set, the default is the value of the server property com.sybase.jaguar.server.external.serverstart.timeout for the server where the component is installed. A value of 0 specifies infinity.

See also — com.sybase.jaguar.component.external.servername, com.sybase.jaguar.server.external.serverstart.timeout

com.sybase.jaguar.component.files

Description — Specifies additional files that are included when the component is archived with the package export feature or replicated with the synchronize feature.

Syntax — Same as for com.sybase.jaguar.applicationclient.files

Usage — The com.sybase.jaguar.component.files property specifies a list of files to be included when the component is exported into a package archive file or replicated to another server with the synchronize feature. By default, the following files are included when you export packages or synchronize between servers:

- The IDL files that define interfaces and types used by the component.
- For C or C++ components, the DLL or shared library that is specified on the General tab of the Component Properties window. If your component requires additional DLLs or shared libraries, you must specify them in the list.

- For Java components the implementation class and any classes listed in the property com.sybase.jaguar.component.java.classes, plus stub classes listed in the com.sybase.jaguar.component.files.corbastubs, and com.sybase.jaguar.component.files.ejbstubs properties.
- For PowerBuilder components, the libraries starting with $ (dollar sign) that are referenced by the property com.sybase.jaguar.component.pb.librarylist.

In EAServer Manager, set this property using the Additional Files tab in the Component Properties dialog box.

Note Java and C++ stubs are not included by default in the component's file set. These can be regenerated on the target server after synchronization or installing the archive. If you do not want to regenerate, add the stub files to the value of the com.sybase.jaguar.component.files property.

See also com.sybase.jaguar.component.java.classes, com.sybase.jaguar.component.files.corbastubs, com.sybase.jaguar.component.files.ejbstubs, com.sybase.jaguar.component.pb.librarylist, com.sybase.jaguar.package.files

com.sybase.jaguar.component.files.corbastubs

Description Specifies the files that implement the component's Java/CORBA stubs.

Syntax Set when Java/CORBA stubs are generated. Do not modify.

See also com.sybase.jaguar.component.files, com.sybase.jaguar.component.files.ejbstubs, com.sybase.jaguar.package.files.corbastubs

com.sybase.jaguar.component.files.ejbstubs

Description Specifies the files that implement the component's EJB stubs.

Syntax Set when EJB stubs are generated. Do not modify.

See also com.sybase.jaguar.component.files, com.sybase.jaguar.component.files.corbastubs, com.sybase.jaguar.package.files.ejbstubs

com.sybase.jaguar.component.generateKey

Description — For an EJB CMP entity bean, or other types of components that use automatic persistence, specifies whether the key values the mapped database table are automatically generated.

Syntax — `true` or `false`. The default is `false`.

See also — "Enabling automatic key generation" in Chapter 27, "Creating Entity Components," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.component.home

Description — Specifies the component's home IDL interface. EJB components must have a home interface, and other component types must have one to support access from EJB clients.

Syntax — *home*

Where *home* is the IDL interface name. For example, examples::QueryHome. For an EJB component, the default is:

package::*component*Home

Where *package* is the EAServer package name, and *component* is the EAServer package name. If the component does not have remote interfaces, set the property to an empty string or remove the setting.

For other component types, there is no default.

Usage — EJB components must have a home interface or a local home interface (specified by com.sybase.jaguar.component.local.home). A component of any type must have a home interface to be invoked from EJB clients or EJB components.

If an EJB component has only local interfaces, that is, no home and remote interface, when you deploy the EJB, EAServer sets the com.sybase.jaguar.component.remote property to match the local IDL interface (com.sybase.jaguar.component.local) and sets the com.sybase.jaguar.component.home property to match the local home IDL interface (com.sybase.jaguar.component.local.home).

In EAServer Manager, set this property using the File menu for the component's Interfaces folder.

See also com.sybase.jaguar.component.remote,
com.sybase.jaguar.component.ejb.home,
com.sybase.jaguar.component.local,
com.sybase.jaguar.component.local.home

com.sybase.jaguar.component.home.ids

Description Specifies repository IDs for the component's home IDL interface. Derived from com.sybase.jaguar.component.home.

Syntax A comma-separated list of repository IDs.

Usage This property is read-only. It is set automatically when you set the com.sybase.jaguar.component.home property. In EAServer Manager, view the property using the Advanced tab in the Component Properties dialog box.

See also com.sybase.jaguar.component.home

com.sybase.jaguar.component.identitytype

Description Specifies whether to use the connection user name or SSL certificate for role-based authorization.

Syntax Allowable values are:

Value	To indicate
jaguar	Authorize based on the EAServer connection user name
ssl	Authorize based on the SSL certificate digital ID

Usage In EAServer Manager, set this property on the Advanced tab in the Component Properties dialog box.

com.sybase.jaguar.component.ids

Description Specifies repository IDs for the component's IDL interfaces. Derived from com.sybase.jaguar.component.interfaces.

Syntax A comma-separated list of repository identifiers.

Usage	This property is read-only. It is set automatically when you set the com.sybase.jaguar.component.interfaces property. In EAServer Manager, view the property using the Advanced tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.interfaces

com.sybase.jaguar.component.instancePool

Description	Constrains the component to run in the specified instance pool.
Syntax	The name of the instance pool. Table B-7 on page 455 describes the default used when this property is not set.
See also	Instance pool properties on page 454

com.sybase.jaguar.component.interfaces

Description	Specifies the IDL interfaces that the component supports for client use.
Syntax	A comma-separated list of IDL interface names. For example: MyModule::iFace1,MyModule::iFace2 The default is: *package*::*component* Where *package* is the package name, and *component* is the component name.
Usage	In EAServer Manager, the component's interfaces can be modified using the Interfaces folder displayed beneath the component icon. See Chapter 5, "Defining Component Interfaces," in the *EAServer Programmer's Guide* for more information.
See also	com.sybase.jaguar.component.remote

com.sybase.jaguar.component.iso_level

Description	For EJB 1.0 components, specifies the isolation level for transactions begun by the component's methods.
Syntax	Table B-1 lists allowable values. You can specify the isolation level for individual methods by setting the method property com.sybase.jaguar.method.iso_level.

Table B-1: Transaction isolation level property values

Value	Transaction isolation level
`undefined` (the default)	If set at the component level, use the database or JDBC driver default setting. If set at the method level, use the component level setting.
`read_uncommitted`	Read uncommitted.
`read_committed`	Read committed.
`repeatable_read`	Repeatable read.
`serializable`	Serializable.

Usage — In EAServer Manager, you can set this property using the Transaction Isolation Level field on the Transactions tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.tx_type, com.sybase.jaguar.method.iso_level

com.sybase.jaguar.component.java.class

Description — For Java components (CORBA and EJB), specifies the name of the Java implementation class.

Syntax — The Java class name. For example, com.sybase.jaguar.sample.GlossaryImpl. There is no default.

Usage — In EAServer Manager, set this property in the Java Class field on the General tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.java.classes

com.sybase.jaguar.component.java.classes

Description — For Java components (CORBA and EJB), specifies additional classes to be loaded by the component class loader.

Syntax — Same as for com.sybase.jaguar.application.java.classes.

Usage — See “Custom class lists for Java and EJB components” in Chapter 30, “Configuring Custom Java Class Lists,” in the *EAServer Programmer’s Guide*.

In EAServer Manager, set this property using the Java Classes tab in the Component Properties dialog box.

See also com.sybase.jaguar.component.java.class, com.sybase.jaguar.component.refresh, com.sybase.jaguar.application.java.classes, com.sybase.jaguar.package.java.classes, com.sybase.jaguar.server.java.classes

com.sybase.jaguar.component.key

Description For an entity component, specifies the IDL datatype for the primary key.

Syntax The value can be one of the following:

- **An IDL structure** The structure should reflect the primary key for the database relation that the entity bean represents. In other words, add a field for each column in the primary key. Define the structure to match the intended Java package and class name. For example, if the Java class is to be foo.bar.PK1, define a new structure PK1 in module foo::bar. See Chapter 5, "Defining Component Interfaces," in the *EAServer Programmer's Guide* for more information.
- **The name of a serializable Java class** Enter the name of a serializable Java class; for example, foo.bar.MyPK.
- **The IDL string type** Use string if the key relation has only a string column. In Java, the mapped primary key is java.lang.String.

The default is an IDL structure named:

module::*remote*Key

Where *module* is the module that contains the remote interface, and *remote* is the unscoped IDL name of the remote interface. For example, samples::MyBeanKey. If the structure does not exist, EAServer creates a structure with this name containing a single long field named value.

Interoperability and key types Use an IDL structure or string if other types of clients besides Java will use the bean.

Usage In EAServer Manager, set this property in the Primary Key field on the Persistence tab in the Component Properties dialog box.

See also com.sybase.jaguar.component.ejb.key, com.sybase.jaguar.component.list

com.sybase.jaguar.component.key.tc

Description	Specifies the type code string for the primary key IDL type. Derived from com.sybase.jaguar.component.key.
Syntax	The type code string.
Usage	This property is read-only. It is set automatically when you set the com.sybase.jaguar.component.key property. In EAServer Manager, view the value on the Advanced tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.key

com.sybase.jaguar.component.keys

Description	For entity components, specifies the name of an IDL type definition (typedef) for a sequence of the component's primary key datatype. This type is used when generating the skeleton and implementation classes for the component.
Syntax	The name of an IDL typedef that specifies a sequence of primary key type. The default is an IDL typedef: *module*::*remote*Keys Where *module* is the module that contains the remote interface, and *remote* is the unscoped IDL name of the remote interface. EAServer defines this typedef as: typedef sequence < ::*key* > *module*::*remote*Keys; Where *key* is the value of the com.sybase.jaguar.component.key property. If you have used PowerJ or the EAServer Manager import feature to import an entity bean, the com.sybase.jaguar.component.keys typedef may use a different naming convention.
Usage	This property is set to the default when you set the com.sybase.jaguar.component.key property, and the IDL typedef is defined if it does not exist already. Set the property manually to use a value other than the default. In EAServer Manager, view and set this property using the Advanced tab in the Component Properties window.
See also	com.sybase.jaguar.component.key, com.sybase.jaguar.component.ejb.key

com.sybase.jaguar.component.list

Description

For an EJB 1.0 entity bean, specifies the IDL datatype for a sequence of the remote interface type. In an entity bean's IDL home interface, methods that return a collection of instances must return the type name specified by this property.

Syntax

The name of an IDL typedef that specifies a sequence of the remote interface type. The default is an IDL typedef:

module::*comp*List

Where *module* is the module that contains the remote interface, and *comp* is the unscoped IDL name of the remote interface. The default typedef is:

typedef sequence < ::*remote* > *module*::*comp*List;

Where *remote* is the value of the com.sybase.jaguar.component.remote property.

Usage

In EAServer Manager, set this property on the Advanced tab in the Component Properties dialog box.

When you manually specify a value for the Primary Key field on the Persistence tab, EAServer Manager sets this property to ***module*::*component***`List` where *module* is the module containing the primary key type, and *component* is the component name. EAServer Manager defines the type if it does not exist, using the following structure:

```
typedef <sequence ri> componentList
```

where *ri* is the remote interface name, and *component* is the component name.

Set the com.sybase.jaguar.component.list property only when you have manually defined a sequence that uses another naming convention or that is located in another module.

If you have used PowerJ or the EAServer Manager import feature to import an entity bean, the com.sybase.jaguar.component.keys typedef may use a different naming convention.

See also

com.sybase.jaguar.component.remote

com.sybase.jaguar.component.listener

Description

For message-driven beans (MDBs), specifies the message listener package and component.

Syntax

The package and component name for the MDB listener; for example:

MDBpackage/MDBcomponent

To specify a thread pool, append the thread pool name in square brackets, for example:

MDBpackage/MDBcomponent[MyThreadPool]

The thread pool must have one or more worker threads. A thread pool with multiple worker threads enables the message listener to process multiple messages at the same time. If you do not specify the name of a thread pool, the message listener uses the <system> default thread pool, which has a single worker thread. You can create thread pools as described in “Thread pools” on page 174.

Usage — In EAServer Manager, specify the Listener on the MDB Type tab in the Component Properties dialog box.

See also — JMS entity properties, “Listeners” on page 171, “Thread pools” on page 174

com.sybase.jaguar.component.load

Description — Obsolete.

Syntax — This property is obsolete.

See also — com.sybase.jaguar.component.store

com.sybase.jaguar.component.local

Description — For EJB 2.0 components, specifies the name of the IDL local interface.

Syntax — *local*

Where *local* is the IDL interface name. For example, examples::LocalQuery. The local interface must be one of the interfaces listed in the com.sybase.jaguar.component.interfaces property. If the component has no local interfaces, set the property to an empty string or remove the setting.

Usage — In EAServer Manager, set this property using the File menu for the Interfaces folder displayed beneath the component icon.

See also — com.sybase.jaguar.component.local.home, com.sybase.jaguar.component.remote, com.sybase.jaguar.component.ejb.local

com.sybase.jaguar.component.local.home

Description	For EJB 2.0 components, specifies the name of the IDL local home interface.
Syntax	*home* Where home is the IDL interface name. For example, examples::LocalQueryHome. If the component has no local interfaces, set the property to an empty string or remove the setting.
Usage	In EAServer Manager, set this property using the File menu for the Interfaces folder displayed beneath the component icon.
See also	com.sybase.jaguar.component.local, com.sybase.jaguar.component.home

com.sybase.jaguar.component.lwc

Description	For EJB components only. Enables the EAServer lightweight container (LWC) for calls to this component from EJBs or servlets and JSPs hosted in the same server. For more information, see "Lightweight container" in the *EAServer Performance and Tuning Guide*.
Syntax	`true` or `false`. A value of `true` enable the LWC for this component. The default is `false` disables LWC.
Usage	To enable LWC for components, the server property com.sybase.jaguar.server.lwc must be true.
See also	com.sybase.jaguar.component.lwc.enableSkeletons, com.sybase.jaguar.server.lwc

com.sybase.jaguar.component.lwc.enableSkeletons

Description	Enables LWC calls to EJB components from servlets and JSPs hosted in the same server. Such calls are not supported unless this option is set..
Syntax	`true` or `false`. A value of `true` enables calls from the Web tier. The default is false. To enable this setting for components, the server property com.sybase.jaguar.server.lwc.enableSkeletons must be true. Regenerate stubs and skeletons for the component after modifying this setting.
See also	com.sybase.jaguar.component.lwc, com.sybase.jaguar.server.lwc.enableSkeletons

com.sybase.jaguar.component.maxpool

Description — When instance pooling is enabled, specifies the maximum pool size.

Syntax — Specify a positive integer value, or 0 to indicate the pool size has no limit. The default is 0, which means no maximum pool size is in effect.

Usage — If the maximum pool size is reached, EAServer destroys excess instances after deactivation. Set this property in EAServer Manager using the Maximum Pooled Instances field on the Resources tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.pooling, com.sybase.jaguar.component.minpool, com.sybase.jaguar.component.objects

com.sybase.jaguar.component.maxwait

Description — This setting applies only when the com.sybase.jaguar.component.objects property is set to specify a limit on the number of simultaneous active instances.

Syntax — *wait_time*

Where *wait_time* is the time to wait, in seconds.

Usage — If a request arrives when the maximum number of instances exist and are all busy, the request blocks, with blocking time constrained by this property. If the blocking time expires, the caller receives a CORBA::NO_RESOURCE_EXCEPTION

In EAServer Manager, set this property using the Maximum Wait field on the Resources tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.objects

com.sybase.jaguar.component.mdb.acknowledge-mode

Description — Specifies the acknowledgment mode for MDBs that manage their own transactions.

Syntax — Allowable values are:

Value	To indicate
`Auto-acknowledge`	AUTO_ACKNOWLEDGE mode; the session automatically acknowledges messages

Value	To indicate
`Dups-ok-acknowledge`	DUPS_OK_ACKNOWLEDGE mode; instructs a session to lazily acknowledge messages, which reduces a session's workload but may lead to duplicate message deliveries

Usage — In EAServer Manager, set the Acknowledge Mode on the MDB Type tab in the Component Properties dialog box.

See also — Chapter 31, "Using the Message Service," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.component.mdb.destination-type

Description — Specifies whether the MDB is associated with a JMS topic or message queue.

Syntax — Allowable values are:

Value	To indicate
`javax.jms.Topic`	Topic
`javax.jms.Queue`	Message queue

Usage — In EAServer Manager, set the Destination Type on the MDB Type tab in the Component Properties dialog box.

com.sybase.jaguar.component.mdb.message-selector

Description — For an MDB associated with a message queue, specifies the message selector for a message queue. The message service uses the selector to filter the message that it delivers to the queue.

Syntax — topic='*topicString*'

Usage — In EAServer Manager, define the Message Selector on the MDB Type tab in the Component Properties dialog box.

See also — "Message selectors" on page 170.

com.sybase.jaguar.component.mdb.subscription-durability

Description — When the MDB is associated with a topic, specifies whether the topic is durable or nondurable.

Syntax

Allowable values are:

Value	To indicate
`Durable`	Durable topic subscriber; guarantees message delivery
NonDurable	Nondurable topic subscriber; can receive published messages only while it is connected to EAServer

Usage

In EAServer Manager, set the Topic Subscription Durability on the MDB Type tab in the Component Properties dialog box.

See also

Chapter 31, "Using the Message Service," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.component.methods

Description

Used to store the component's nondefault method properties.

Syntax

Undocumented. Use the Method entity type and the com.sybase.jaguar.method.* property names to set and retrieve method properties.

See also

Method properties on page 480

com.sybase.jaguar.component.minpool

Description

When instance pooling is enabled, specifies the minimum pool size.

Syntax

Specify a 0 or a positive integer value. The default is 0.

Usage

Set this property in EAServer Manager using the Minimum Pooled Instances field on the Resources tab in the Component Properties dialog box.

See also

com.sybase.jaguar.component.pooling, com.sybase.jaguar.component.maxpool

com.sybase.jaguar.component.model

Description

Specifies the component model.

Syntax

Allowable values are:

Value	To indicate
`ejb`	Enterprise JavaBeans model.

Value	To indicate
`cts`	CORBA with control interface CtsComponents::ObjectControl
`mts`	ActiveX components and CORBA components that use a control interface other than CtsComponents::ObjectControl.
(none)	No value required for component types not mentioned above.

Usage — EAServer Manager sets this property correctly when you change the component type. To view the property, use the Advanced tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.control, com.sybase.jaguar.component.model.version, com.sybase.jaguar.component.type

com.sybase.jaguar.component.model.version

Description — For an EJB component, specifies the EJB specification version.

Syntax — Allowable values are “1.0”, “1.1”, and “2.0”. The default is “2.0”.

Usage — In EAServer Manager, set this property using the EJB Version control on the General tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.model, com.sybase.jaguar.component.type

com.sybase.jaguar.component.monitor

Description — Assigns this component to a thread monitor.

Syntax — Specify the monitor name.

Usage — If you change this setting, you must regenerate and recompile the component skeleton for the change to take effect.

See also — Thread monitor properties, com.sybase.jaguar.method.monitor

com.sybase.jaguar.component.monitor.MaxRespTime

Description	Specifies the maximum allowable average response time for the component. If the average method completion time rises above this limit, EAServer blocks creation of additional instances of this component until the average drops below the specified limit.
Syntax	The maximum allowable average response time for the component, in seconds. The default is -1, which indicates no time limit.
See also	com.sybase.jaguar.component.monitor.MinInstance Chapter 9, "Using the Performance Monitor," in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.component.monitor.MinInstance

Description	When response time monitoring is in effect (com.sybase.jaguar.component.monitor.MaxRespTime is set to a non-negative number), specifies the minimum numer of instances that must be allowed to execute regardless of observed response times.
Syntax	The minimum numer of instances. The default is -1, which means no new instances are blocked by the Performance Monitor.
See also	com.sybase.jaguar.component.monitor.MaxRespTime Chapter 9, "Using the Performance Monitor," in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.component.name

Description	Specifies the name of the component and the package in which it is installed.
Syntax	*package*/*component* Where: • *package* is the package in which the component is installed. • *component* is the name of the component.
Usage	In EAServer Manager, the component name is set when defining a new component and cannot be changed.

com.sybase.jaguar.component.objectCache

Description: Specifies the name of the component that implements object caching for entity components that use automatic persistence and EJB CMP entity beans.

Syntax: The component name, in the form *package*/*component*. To use the built-in implementation, enter:

```
CtsComponents/ObjectCache
```

See also: com.sybase.jaguar.component.objectCache.sizeCheckInterval

com.sybase.jaguar.component.objectCache.sizeCheckInterval

Description: For EJB CMP entity beans that use the object cache, specifies how often EAServer checks the size of the entry before placing it in the cache.

Syntax: A positive integer. The default is 1. If you specify a size check interval *N*, EAServer performs the size calculation on only every *N*th entry, and uses a running average size for the interim entries.

Usage: When writing to the object cache, EAServer checks the size of the entry to see if the cache size would be exceeded. If so, least recently used entries are flushed from the cache until there is room for the new entry. The time spent calculating size can adversely affect performance. However, if the size of the data varies a lot, setting a size check interval may lead to inaccurate cache size estimations, resulting in memory use beyond the configured cache size.

See also: com.sybase.jaguar.component.objectCache

com.sybase.jaguar.component.objects

Description: Specifies the maximum number of component instances that can exist at once. For a C++ component that runs as an external process, specifies the maximum number of simultaneously running external processes.

Syntax: *n*

Where *n* is a positive integer.

Usage: This property specifies the maximum number of instances that can be created at once. If a request arrives when the maximum number of instances exist and are all busy, the request blocks, with blocking time constrained by the com.sybase.jaguar.component.maxwait property.

In EAServer Manager, set this property in the Maximum Active Instances field on the Resources tab in the Component Properties dialog box.

See also: com.sybase.jaguar.component.cpp.process, com.sybase.jaguar.component.maxwait, com.sybase.jaguar.component.maxpool

com.sybase.jaguar.component.passByReference

Description

Enables the proprietary EJB pass-by-reference in-server invocation mechanism supported by some other J2EE vendors.

Syntax

`true` to enable pass-by-reference semantics or `false` to disable. The default is `false`.

Usage

When this property is set to true, EJB stubs and skeletons for the component and its home and remote interfaces use the same parameter passing mode that EAServer normally uses for EJB 2.0 local interfaces. After changing the value, you must regenerate stubs and skeletons.

This feature is not intended for new development, which should use standard EJB 2.0 local interfaces. When used, remote clients will be unable to call the component. The feature cannot be used with components that already have a local interface. If two or more components share the same home and remote interfaces, then all or none of those components must be configured for pass-by-reference.

See also

"Optimizing in-server EJB calls" in Chapter 3, "Component Tuning," in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.component.pb.appname

Description

For PowerBuilder components, the name of the PowerBuilder application that contains the NVO that implements the component.

Syntax

The name of the PowerBuilder application, as displayed in PowerBuilder.

See also

com.sybase.jaguar.component.pb.class

com.sybase.jaguar.component.pb.class

Description	For PowerBuilder components, specifies the name of the nonvisual object that implements the component's methods.
Syntax	The name of the PowerBuilder NVO, as displayed in PowerBuilder.
See also	com.sybase.jaguar.component.pb.appname

com.sybase.jaguar.component.pb.cookie

Description	For versioned PowerBuilder components, contains version cookie data.
Syntax	The cookie data.
See also	com.sybase.jaguar.component.pb.appname, com.sybase.jaguar.component.pb.class

com.sybase.jaguar.component.pb.debug

Description	For PowerBuilder components, specifies whether the component can be debugged while executing.
Syntax	`true` to enable debugging, or `false` to disable debugging. The default is `false`.
See also	com.sybase.jaguar.component.debug

com.sybase.jaguar.component.pb.librarylist

Description	For PowerBuilder components, specifies library files that are required to run the object.
Syntax	The list of library files, separated by semicolons. Prefix library names with a dollar sign ($) if they must be included when the component is synchronized or archived. For example: mylib.pbl;anotherlib.pbl;$utils.pbl
See also	com.sybase.jaguar.component.files, com.sybase.jaguar.component.pb.appname, com.sybase.jaguar.component.pb.class

com.sybase.jaguar.component.pb.live_edit

Description	For PowerBuilder components, enables or disables the live editing feature.
Syntax	`true` to enable live editing; `false` to disable. The default is `false`.
Usage	This property should be set from the PowerBuilder user interface, as the live editing feature depends on corresponding changes to the component's project settings. See the PowerBuilder documentation for more information.
See also	com.sybase.jaguar.component.pb.debug

com.sybase.jaguar.component.pb.version

Description	For PowerBuilder components, specifies the required version of the PowerBuilder virtual machine.
Syntax	This property is set when deploying from PowerBuilder, and should not be edited in any other way. Components that lack this property setting are run in the version 7.0 VM.
See also	com.sybase.jaguar.component.pb.appname, com.sybase.jaguar.component.pb.class

com.sybase.jaguar.component.pooling

Description	Specifies whether component instances are pooled for reuse by multiple client sessions.
Syntax	`true` or `false`. The default is `true`.
Usage	When this property is true, component instances are always pooled after deactivation. For components that use the JaguarEJB::ServerBean or JaguarCOM::ObjectControl control interfaces, you can also configure pooling by implementing interfaces with a canReuse (ServerBean) or canBePooled (JaguarCOM) method. If you enable the Pooling option in EAServer Manager, your component is always pooled, and these methods are not called. In EAServer Manager, set this property with the Pooling check box on the Instances tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.control, com.sybase.jaguar.component.maxpool, com.sybase.jaguar.component.minpool

com.sybase.jaguar.component.ps

Description — For an entity or stateful session component, specifies how persistence is performed.

Syntax — Allowable values are:

Value	To indicate
component	The implementation manages persistence, via the ejbLoad and ejbStore methods, or ctsLoad and ctsStore.
automatic	The server manages persistence with code generated as part of the component skeleton, using mapping properties to map implementation fields to database table columns.
generated	A generated class (specified by the com.sybase.jaguar.component.ps.class property) manages persistence. The recommended setting for EJB entity beans that require container-managed persistence.
serialize	For Java components only. The component state is serialized and stored in binary form.
none	No persistence.

Usage — In EAServer Manager, set this property using the Persistence Type control on the Persistence tab in the Component Properties dialog box.

See also — Chapter 31, "Using the Message Service," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.component.ps.class

Description — For entity components, specifies the generated class to handle component persistence when com.sybase.jaguar.component.ps property is "generated."

Syntax — The Java class name.

Usage — In EAServer Manager, set this property in the Generated Class field on the Persistence tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.ps

com.sybase.jaguar.component.qop

Description — Minimum quality of protection required to access component.

Syntax	A QOP identifier. “Securing Component Access,” in the *EAServer Security Administration and Programming Guide* lists allowable QOP values.
Usage	In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.

com.sybase.jaguar.component.queue

Description	For MDBs associated with a message queue, specifies the name of the message queue.
Syntax	*MyQueue*
Usage	In EAServer Manager, specify the Name on the MDB Type tab in the Component Properties dialog box.
See also	“Permanent destinations” on page 165, Chapter 31, “Using the Message Service,” in the *EAServer Programmer’s Guide*

com.sybase.jaguar.component.reentrant

Description	For entity components, including EJB entity beans, specifies whether a component can recursively call itself, or participate in an intercomponent call sequence that creates recursion.
Syntax	`true` or `false`. The default is `false`.
Usage	In EAServer Manager, set this property with the Reentrant check box on the Instances tab in the Component Properties dialog box.

com.sybase.jaguar.component.refresh

Description	Specifies whether the component can be refreshed, that is, whether a new implementation can be loaded while the server is running.
Syntax	`true` or `false`. The default is `true`.
Usage	For Java components, disabling the refresh property causes the component and dependent classes to be loaded by the system class loader rather than being custom-loaded. For more information, see these sections in the *EAServer Programmer’s Guide*: For EJB components,

- "Disabling refresh" in Chapter 7, "Creating Enterprise JavaBeans Components"
- For CORBA Java comonents, "Disabling component refresh" in Chapter 11, "Creating CORBA Java Components"

In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.

See also

com.sybase.jaguar.component.cpp.copy,
com.sybase.jaguar.component.java.classes,
com.sybase.jaguar.package.java.classes,
com.sybase.jaguar.application.java.classes,
com.sybase.jaguar.server.java.classes

com.sybase.jaguar.component.remote

Description

Specifies the name of the remote (main) IDL interface. EJB components must have a remote interface, and other component types must have one to support access from EJB clients.

Syntax

remote

Where *remote* is the IDL interface name. For example, examples::Query. The remote interface must be one of the interfaces listed in the com.sybase.jaguar.component.interfaces property. For an EJB component, the default is set to:

package::*component*

Where *package* is the EAServer package name, and *component* is the EAServer package name. If the component does not have remote interfaces, set the property to an empty string or remove the setting.

For other component types, there is no default.

Usage

In EAServer Manager, set this property using the File menu for the Interfaces folder displayed beneath the component icon.

If an EJB component has only local interfaces, that is, no home and remote interface, when you deploy the EJB, EAServer sets the com.sybase.jaguar.component.remote property to match the local IDL interface (com.sybase.jaguar.component.local) and sets the com.sybase.jaguar.component.home property to match the local home IDL interface (com.sybase.jaguar.component.local.home).

See also com.sybase.jaguar.component.home, com.sybase.jaguar.component.ejb.remote, com.sybase.jaguar.component.local, com.sybase.jaguar.component.interfaces

com.sybase.jaguar.component.resource-env-ref

Description Resource environment references are logical names applied to objects administered by EAServer.

Syntax See Resource environment reference properties on page 489.

Usage In EAServer Manager, set this property using the Resource Env Refs tab in the Component Properties dialog box.

See also Resource environment reference properties on page 489.

com.sybase.jaguar.component.resource-ref

Description For EJB components, specifies aliased JNDI names for database connections, JavaMail sessions, and URL factories used by the component.

Syntax See Resource reference properties on page 496.

Usage In EAServer Manager, set this property using the Resource Refs tab in the Component Properties dialog box.

See also Resource reference properties on page 496.

com.sybase.jaguar.component.retry.timeout

Description For an EJB MDB component, specifies how long the server should redeliver a message after the MDB has thrown an exception while processing the message.

Syntax Specify the time period in seconds. The default is zero, which indicates the message should not be redelivered. EAServer does not retry unless com.sybase.jaguar.component.auto.failover is set to `true`.

Usage To enable redelivery of messages after your MDB throws an exception, you must specify a retry timeout by setting this property and setting com.sybase.jaguar.component.auto.failover is set to `true`. EAServer retries at intervals that increase by about one second after each failure; that is, after one second, again after 3 seconds, again after 6 seconds, and so forth.

See also com.sybase.jaguar.component.auto.failover

com.sybase.jaguar.component.roles

Description For component types other than EJB, specifies roles that a user must belong to invoke the component.

Syntax *role1*, *role2*, ...

Where *role1*, *role2*, and so forth are role names defined in the repository.

Usage In EAServer Manager, set this property using the File menu for the Roles folder displayed below the component icon.

Roles are attached to EAServer packages, components, and methods. Attaching a role to a package controls access to all non-EJB components in the package. Attaching a role to a component constrains access to all methods in the component's interfaces. Attaching a role to a method constrains access to that method.

For EJB components, configure access control at the method level by setting the method property com.sybase.jaguar.method.security-roles for each method.

See also com.sybase.jaguar.package.roles,
com.sybase.jaguar.method.roles,
com.sybase.jaguar.method.security-roles

com.sybase.jaguar.component.runasidentity

Description For EJB 1.0 components, specifies an identity name used for intercomponent calls if the com.sybase.jaguar.component.runasmode property is "specified."

Syntax *id*

Where *id* is the alias name for an identity. The name used must be aliased to an entity that exists in the EAServer repository by setting a com.sybase.jaguar.package.runasidentity.<id> property for the package that contains this component. For example, if the component's run-as identity property is "ejbFoo", create a package property with this name and value:

```
com.sybase.jaguar.package.runasidentity.ejbFooId=fooId
```

Usage In EAServer Manager, set this property using the Run As Identity field on the Run As Mode tab in the Component Properties dialog box.

See also com.sybase.jaguar.component.runasmode, com.sybase.jaguar.package.runasidentity.<id>

com.sybase.jaguar.component.runasmode

Description For EJB 1.0 components, specifies the user identity that is assumed for intercomponent calls.

Syntax Allowable values are:

Value	Description
client	Use the client's user name and password. This is the default setting.
system	Use the system user name and password. The system account effectively belongs to any role, and can execute any method on any component that is installed in the same server or cluster.
specified	Use the user name and password associated with the identity name specified by the com.sybase.jaguar.component.runasidentity property.

Usage In EAServer Manager, set this property using the Run As Mode tab in the Component Properties dialog box.

See also com.sybase.jaguar.component.runasidentity, com.sybase.jaguar.method.runasmode

com.sybase.jaguar.component.SAXfactory

Description Specifies the class name for a custom SAX XML parser factory class.

Syntax The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.component.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.

See also com.sybase.jaguar.component.DOMfactory, com.sybase.jaguar.application.XSLTfactory

com.sybase.jaguar.component.security.runasidentity

Description For EJB 2.0 components, specifies the run-as identity used for intercomponent calls. If this property is not set, intercomponent calls use the client identity.

Syntax

(specified=*id*,role=*role-name*,desc=*desc*)

Where:

Variable	Specifies
id	Is a logical identity name which must be mapped to an EAServer identity by setting a corresponding package property, com.sybase.jaguar.package.runasidentity.<id> where `<id>` is the logical identity name.
role-name	Is a logical role name which must be mapped to an EAServer role by setting a corresponding package property, com.sybase.jaguar.package.security-role.<j2ee-role> where `<j2ee-role>` is the logical role name. The mapping can also be established by setting the com.sybase.jaguar.application.security-role.<j2ee-role> property. The mapped EAServer identity should be in the mapped EAServer role.
desc	Is an optional description of the run-as authorization requirement. The description can help users when the component is deployed to another server, and the deployer must choose a different identity mapping.

See also

com.sybase.jaguar.package.runasidentity.<id>,
com.sybase.jaguar.package.security-role.<j2ee-role>,
com.sybase.jaguar.application.security-role.<j2ee-role>,
com.sybase.jaguar.component.runasidentity

com.sybase.jaguar.component.security-role-ref.<j2ee-role-ref>

Description

For EJB components, maps a role reference specified in the com.sybase.jaguar.component.security-role-refs property to a J2EE role name defined in package or application properties.

Syntax

com.sybase.jaguar.component.security-role-ref.*role-ref=j2ee-role*

Where:

- *j2ee-role* is a J2EE role name defined for the package or application that contains the component, using the com.sybase.jaguar.application.security-roles or com.sybase.jaguar.package.security-roles property, respectively.
- *role-ref* is a role name defined in the com.sybase.jaguar.component.security-role-refs property.

Usage	Applies to EJB (1.1 or 2.0) components only. In EAServer Manager, set this property on the Role Refs tab in the Component Properties dialog box.
See also	com.sybase.jaguar.application.security-roles, com.sybase.jaguar.package.security-roles

com.sybase.jaguar.component.security-role-refs

Description	For EJB components, specifies role reference names used within isCallerInRole method invocations in the component implementation.
Syntax	A comma-separated list of role reference names. For each role reference specified by this property, you must map the role reference to a J2EE role name by setting the corresponding com.sybase.jaguar.component.security-role-ref.<j2ee-role-ref> property.
Usage	Applies to EJB (1.1 or 2.0) components only. Role references are required if the component implementation calls the isCallerInRole Java method to programmatically restrict access. In EAServer Manager, set this property on the Role Refs tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.security-role-ref.<j2ee-role-ref>

com.sybase.jaguar.component.selectForUpdate

Description	For components that use automatic persistence, such as EJB CMP entity beans, specifies whether queries create exclusive locks on selected rows.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.component.touchColumn, com.sybase.jaguar.component.ps, com.sybase.jaguar.component.selectWithLock, com.sybase.jaguar.component.timestamp

com.sybase.jaguar.component.selectWithLock

Description	For components that use automatic persistence, such as EJB CMP entity beans, specifies whether queries create shared locks on selected rows.

Syntax	`true` or `false`. The default is `false`.
Usage	This property or the com.sybase.jaguar.component.selectForUpdate property enable pessimistic concurrency control, that is, table or row locking to avoid concurrent updates to the same row. The alternative model uses optimistic concurrency control, using value comparison to roll back updates if the selected rows have been updated by another user. In EAServer Manager, set this property using the Select With Lock option on the Persistence/General tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.ps, com.sybase.jaguar.component.selectForUpdate, com.sybase.jaguar.component.timestamp

com.sybase.jaguar.component.sharing

Description	Specifies whether multiple component instances can be created. A *shared* component serves all client requests with one instance.
Syntax	`true` or `false`. The default is `false`.
Usage	When this option is enabled, a single, shared instance of the component services all client requests. A shared component can store data in instance variables. However, if the com.sybase.jaguar.component.thread.safe property is true, you must synchronize access to instance variables. **Sharing setting overrides Pooling setting** If both com.sybase.jaguar.component.sharing and com.sybase.jaguar.component.pooling are enabled (true), the sharing property takes precedence. In EAServer Manager, set this property using the Sharing check box on the Instances tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.pooling, com.sybase.jaguar.component.thread.safe

com.sybase.jaguar.component.softLock

Description	For EJB CMP entity beans that use an isolation level of `repeatable_read_with_cache`, enables soft locking.

Syntax — `true` or `false`. `true` enables soft locking. The default of `false` disables soft locking.

See also — com.sybase.jaguar.component.cmp_iso_level, com.sybase.jaguar.component.softLock.timeout, "Configuring CMP isolation level" in the *EAServer Performance and Tuning Guide*

com.sybase.jaguar.component.softLock.timeout

Description — When soft locking is enabled, specifies the timeout period for soft-locked rows.

Syntax — The timeout, in seconds.

See also — com.sybase.jaguar.component.softLock, "Configuring CMP isolation level" in the *EAServer Performance and Tuning Guide*

com.sybase.jaguar.component.state

Description — Specifies the datatype for component state information. Applies to entity components or stateful components that use automatic storage other than EJB entity beans and stateful session beans.

Syntax — The name of an IDL structure or serializable Java class that stores component state.

Usage — In EAServer Manager, set this property in the State field on the Persistence tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.ps, Chapter 27, "Creating Entity Components," and Chapter 28, "Configuring Persistence for Stateful Session Components," in the *EAServer Programmer's Guide*

com.sybase.jaguar.component.state.gs

Description — For entity components or stateful components that use automatic persistence, specifies component methods that set and retrieve component state.

Syntax — For EJB entity beans and stateful session beans, set the value to `default`. For other stateful component types, set the value to:

getMethod, *setMethod*

Where *getMethod* is the name of the method that returns component state information, and *setMethod* is the name of the method that receives component state information and applies it to instance fields. If you specify no value, the default is `getState,setState`. Your component implementation must contain these methods, but they should not be listed in the component's client interfaces. The getState method returns an instance of the type specified by the State field, and the setState method accepts a parameter of this type. For example, if the state type is ShoppingCartState, the getState and setState methods might be defined as follows in Java:

```
private ShoppingCartState data;

ShoppingCartState getState()
{
    return data;
}

void setState(ShoppingCartState state)
{
    data = state;
}
```

Usage — In EAServer Manager, set this property in the State Methods field on the Persistence tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.ps, Chapter 27, "Creating Entity Components," and Chapter 28, "Configuring Persistence for Stateful Session Components," in the *EAServer Programmer's Guide*

com.sybase.jaguar.component.stateless

Description — For EJB session beans and non-EJB components that use the control interface CtsComponents::ObjectControl, specifies whether the component is stateless, that is, whether an instance should be bound to a client session for its lifetime.

For EJB session beans, this property is set automatically when the component type is set, and must not be changed. For other component types, the option must be set manually.

Syntax — `true` or `false`. The default is `false`.

Usage — In EAServer Manager, set this property using the Stateless check box on the Instances tab in the Component Properties window.

See also com.sybase.jaguar.component.control, com.sybase.jaguar.component.tx_vote

com.sybase.jaguar.component.storage

Description For entity or stateful session components that use automatic persistence or failover support, specifies the storage component.

Syntax For a stateful session component that uses in-memory state replication, specify:

CtsComponents/HeapStorage

For an entity component or stateful session component that uses persistent (remote database) storage, specify:

storage(cache=*cacheName*,table=*tableName*)

For an EJB stateful session between configured for passivation, specify:

storage(cache=*cacheName*,table=*tableName*,transient)

Where:

- *storage* is the name of the storage component.
- *cacheName* is the name of the connection cache to use. The default for new components is `JavaCache`. For components that are imported from an EJB-JAR file, the default is the value of the server property com.sybase.jaguar.server.defaultStorageCache.
- *tableName* is the name of the database table to store component state. If using mapped fields, prefix the table name with "map:", as in "map:mytable."

In EAServer Manager, set this property in the Storage Component field on the Persistence tab in the Component Properties dialog box.

See also com.sybase.jaguar.component.ps, com.sybase.jaguar.component.timestamp, Chapter 27, "Creating Entity Components," and Chapter 28, "Configuring Persistence for Stateful Session Components," in the *EAServer Programmer's Guide*

com.sybase.jaguar.component.store

Description Tells the server when to call ejbStore.

Syntax

store

Where *store* can be one of, or a comma-separated list of more than one of, these values:

Value	Description
`afterCreate`	The server calls ejbStore after it calls ejbPostCreate.
`afterInvoke`	The server calls ejbStore after it calls each business method. `afterInvoke` and `beforeCompletion` cannot be used together.
`beforeCompletion`	The server calls ejbStore at the end of a transaction. This option may provide the best performance, but you may get the wrong result in some cases; for example, if two entity beans in one transaction are updating the same table, or if updates from session and entity beans are mixed in one transaction. `afterInvoke` and `beforeCompletion` cannot be used together.

`afterInvoke` and `beforeCompletion` cannot be used together. The default value is `afterCreate,afterInvoke`.

Usage

The default value of `afterCreate,afterInvoke` is required for EJB 2.0 compliance and is safe for all compliant entity bean implementations

If you insert values in the ejbCreate method, and do not modify any field values in the ejbPostCreate method, you can safely remove `afterCreate` from the setting. Doing so improves performance by eliminating redundant updates to the database.

You can use the `beforeCompletion` setting rather than `afterInvoke` if all updates to one table come from one entity bean, and you do not mind if finder methods return stale values because updates are deferred during a transaction. While this setting yields the best performance, you may get the wrong result with architectures where more than one component can update a table, for example, if two entity beans in one transaction update the same table, or if updates from session and entity beans are mixed in one transaction.

com.sybase.jaguar.component.sync

Description

Specifies whether the component is included in the files replicated by a synchronization operation.

Syntax

`true` or `false`. The default is `true`.

Usage	This property is used to prevent unnecessary replication of built-in EAServer components. You can set it on your own components; however, there is no good reason not to synchronize application components.
See also	Chapter 6, “Clusters and Synchronization”

com.sybase.jaguar.component.thread.safe

Description	Specifies whether multiple component instances can execute concurrently, or whether a shared component can execute simultaneously on multiple threads. If this option is disabled, the server serializes all invocations of component methods.
Syntax	`true` or `false`. The default is `true`.
Usage	Enable this option if the component is thread safe. If the com.sybase.jaguar.component.sharing and com.sybase.jaguar.component.bind.thread properties are both true, the com.sybase.jaguar.component.thread.safe property is implicitly set to false. In EAServer Manager, set this property using the Concurrency check box on the Instances tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.bind.thread, com.sybase.jaguar.component.sharing

com.sybase.jaguar.component.timeout

Description	Specifies how long, in seconds, an active component instance can remain idle between method calls before the client’s proxy becomes invalid.
Syntax	*time* Where *time* is the timeout period in seconds. A value of “0” indicates an infinite timeout. The default is specified by the com.sybase.jaguar.server.timeout server property.
Usage	If the timeout expires, the instance is automatically deactivated. Instance Timeout is useful for ensuring timely deactivation of stateful components. The setting has no effect for stateless components.

When the timeout period is exceeded, EAServer deactivates the component and invalidates the client's object reference. If the client attempts another method invocation, the client-side ORB throws the CORBA::OBJECT_NOT_EXIST exception. At this point, the client must create a new proxy instance for the component.

When specifying timeouts, a resolution of 5 seconds is recommended.

Network transport time is not included in the measured timeout period. You may need to configure a larger timeout period if clients connect over slow networks.

In EAServer Manager, set this property using the Instance Timeout field on the Resources tab in the Component Properties dialog box.

See also: com.sybase.jaguar.server.timeout

com.sybase.jaguar.component.timestamp

Description: For entity components that use automatic persistence with mapped table fields, specifies how the server uses optimistic concurrency control to prevent overlapping updates to the same column.

Syntax: See "Configuring concurrency control" in Chapter 27, "Creating Entity Components," in the *EAServer Programmer's Guide*.

Usage: In EAServer Manager, set this property using the Timestamp field on the Persistence tab in the Component Properties dialog box.

See also: com.sybase.jaguar.component.ps, com.sybase.jaguar.component.selectWithLock com.sybase.jaguar.component.ts.length, com.sybase.jaguar.component.tx_retry

com.sybase.jaguar.component.tlc.sort

Description: For EJB CMP entity beans, specifies that transaction local cache entries for this component should be sorted before ejbStore is called.

Syntax: `true` or `false`. The default of `false` indicates no sorting.

Usage: Setting this property to true helps to avoid deadlock when separate transactions concurrently update multiple instances of the same component.

You cannot enable sorting unless the primary key class implements the java.lang.Comparable interface. Most java.lang utility classes implement this interface, such as String, Integer and so forth.

See also: "Configuring transaction local cache settings" in Chapter 4, "EJB CMP Tuning," in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.component.topic

Description: For MDBs associated with a message topic, specifies the name of the topic.

Syntax: *MyTopic*

Usage: In EAServer Manager, specify the Name on the MDB Type tab in the Component Properties dialog box.

See also: "Permanent destinations" on page 165,
Chapter 31, "Using the Message Service," in the *EAServer Programmer's Guide*

com.sybase.jaguar.component.touchColumn

Description: Used when the com.sybase.jaguar.component.selectForUpdate property is true. For databases such as Sybase Adaptive Server Enterprise that do not support select for update locking syntax, EAServer locks rows by issuing a no-change update statement. This property specifies which column to update.

Syntax: Specify the column name. If you do not specify a column, the first non-key column is updated. For best performance, specify the column with the datatype that can be updated most quickly. For example, int columns can be updated more quickly than varchar columns.

See also: com.sybase.jaguar.component.selectForUpdate

com.sybase.jaguar.component.trace

Description: Enables additional debug tracing for the component, such as logging method names, client IP addresses, and user names for each invocation of a business method.

Syntax: `true` or `false`. The default is `false`.

Usage	In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.debug

com.sybase.jaguar.component.transient

Description	Applies to stateful components only. Specifies whether instances can be run on multiple servers in a cluster or survive a server restart.
Syntax	`true` or `false`. The default is `true`.
Usage	If this option is enabled, the server guarantees that proxy references can only be used within the same server process. For EJB stateful session beans, this property must be enabled for the standard EJB passivation and activation to occur. It must be disabled if you want to configure a stateful session bean to support failover. In EAServer Manager, set this property using the Transient check box on the Instances tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.ps

com.sybase.jaguar.component.ts.length

Description	When com.sybase.jaguar.component.timestamp specifies a column name, this property specifies the datatype and length.
Syntax	No value indicates the default, 4-byte integer. Specify `binary(16)` to use globally unique 16-byte ID timestamps.
Usage	In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.
See also	com.sybase.jaguar.component.ps, com.sybase.jaguar.component.timestamp, Chapter 27, “Creating Entity Components,” in the *EAServer Programmer’s Guide*.

com.sybase.jaguar.component.ts.triggers

Description — For EJB CMP entity beans that use a table-level timestamp, specifies whether EAServer must create triggers to update the timestamp when a row in the mapped table is inserted, modified, or deleted.

Syntax — `true` or `false`. The default is `false`.

See also — com.sybase.jaguar.component.timestamp,
Chapter 27, "Creating Entity Components," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.component.tx_control

Description — Specifies whether control interface methods execute in the context of a server-managed transaction.

Syntax — `true` or `false`. The default is `true` for EJB entity beans, and `false` for other component types.

Usage — In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.control

com.sybase.jaguar.component.tx_outcome

Description — Determines whether a CORBA::TRANSACTION_ROLLEDBACK exception is thrown to the client when a transaction is rolled back.

Syntax — The allowable settings are:

Setting	Description
always	The default. EAServer sends a CORBA::TRANSACTION_ROLLEDBACK exception to the client when a transaction is rolled back.
failed	EAServer does not send the CORBA::TRANSACTION_ROLLEDBACK exception to the client when a transaction is rolled back. If you use this setting, you can code your components to raise a different exception with a descriptive message after calling the RollbackWork transaction primitive. With this setting in effect, EAServer may still throw a CORBA system exception if unable to commit a transaction at your component's request.

Usage

Set this property to `failed` if you require that no exceptions are sent to the client for rolled-back transactions or that an alternate exception is sent. This setting is useful if you require that the client be able to retrieve output parameters after a transaction is rolled back: if an exception is thrown, the output parameters are not available.

In EAServer Manager, set this property using the Advanced tab in the Component Properties dialog box.

com.sybase.jaguar.component.tx_retry

Description

For EJB CMP entity beans, determines whether EAServer automatically replays transactions that fail.

Syntax

`true` or `false`. The default is the value of the server property com.sybase.jaguar.server.tx_retry, or false if the server property is not set.

Usage

This setting applies only to EJB CMP entity beans that use optimistic concurrency control.

EAServer can automatically retry transactions that are rolled back—method calls back to the last transaction boundary are replayed by the server component dispatcher. This feature is useful for For EJB CMP entity beans and entity components that use automatic persistence and optimistic concurrency control. When transaction retry is enabled, transactions that fail due to conflicting updates are automatically retried (replayed).

Auto-retry must be configured for the component that initiates a transaction, which is typically a session bean in EJB applications. Auto-retry works only for intercomponent calls, not for direct invocations of entity beans from the Web tier or base clients.

If retry is enabled, OCC failures throw a CORBA TRANSIENT exception rather than TRANSACTION_ROLLEDBACK. The stub code retries transactions that fail with a CORBA::TRANSIENT exception, up to 10 times.

Auto-retry is not appropriate for all applications. For example, an end user may want to cancel a purchase if the item price has risen. If auto-retry is disabled, clients must be coded to retry or abort transactions that fail because of stale data. The exception thrown is CORBA::TRANSIENT (for EJB clients, this exception is the root cause of the java.rmi.RemoteException thrown by the EJB stub).

See also — com.sybase.jaguar.server.tx_retry,
"Concurrency control options" in Chapter 4, "EJB CMP Tuning," in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.component.tx_timeout

Description — Specifies the maximum duration of an EAServer transaction begun by the component.

Syntax — The timeout period, in seconds, with 0 indicating no timeout. EAServer determines the transaction timeout period as follows:

- If the component Transaction Timeout property is set to a nonzero value, this is the timeout period.
- Otherwise, the server transaction timeout property com.sybase.jaguar.server.tx_timeout is checked. If the server transaction timeout is nonzero, this specifies the timeout period.
- Otherwise, the component Instance Timeout propety com.sybase.jaguar.component.timeout is checked. If this value is nonzero, it specifies the transaction timeout period as well as the instance timeout period.
- Otherwise, the transaction timeout is infinite.

Usage — When specifying timeouts, Sybase recommends a resolution of 5 seconds. EAServer checks for timeouts after each method returns. Your component will not be deactivated in the middle of an invocation because of a timeout. When a transaction times out, the next method invocation in the client-side ORB throws the CORBA::TRANSACTION_ROLLEDBACK system exception.

In EAServer Manager, set this property in the Transaction Timeout field on the Resources tab in the Component Properties dialog box.

See also — com.sybase.jaguar.component.timeout,
com.sybase.jaguar.server.tx_timeout

com.sybase.jaguar.component.tx_type

Description — Specifies how methods in your component participate in transactions.

Syntax — Table B-2 lists allowable values:

Table B-2: tx_type values

Value	To indicate
not_supported	The component-level default. The component's methods never execute as part of a transaction. If the component is activated by another component that is executing within a transaction, the new instance's work is performed outside of the existing transaction.
supports	The component can execute in the context of an EAServer transaction, but a connection is not required to execute the component's methods. If the component is instantiated directly by a base client, EAServer does not begin a transaction. If component A is instantiated by component B, and component B is executing within a transaction, component A executes in the same transaction.
requires	The component always executes in a transaction. When the component is instantiated directly by a base client, a new transaction begins. If component A is activated by component B, and B is executing within a transaction, then A executes within the same transaction; if B is not executing in a transaction, then A executes in a new transaction.
requires_new	Whenever the component is instantiated, a new transaction begins. If component A is activated by component B, and B is executing within a transaction, then A begins a new transaction that is unaffected by the outcome of B's transaction; if B is not executing in a transaction, then A executes in a new transaction.
mandatory	Methods may only be invoked by a client that has an outstanding transaction.

Value	To indicate
user_managed	The component implementation explicitly begins and ends transactions. **For EJB session bean components** The component can explicitly begin, commit, and roll back new, independent transactions by using the javax.transaction.UserTransaction interface. Transactions begun by the component execute independently of the client's transaction. If the component has not begun a transaction, the component's database work is performed independently of any EAServer transaction. Stateless session beans can use this attribute, but transactions begun in a method must be committed or rolled back before that method returns. Otherwise, EAServer logs an error and returns an exception to the client. Stateful session beans can create transactions that remain open across several method calls. **For components of other types** The component can inherit a client's transaction. If called without a transaction, the component can explicitly begin, commit, and roll back transactions by using the CORBA CosTransactions::Current interface.
never	The component's methods never execute as part of a transaction, and the component cannot be called in the context of a transaction. If a client or another component calls the component with an outstanding transaction, EAServer throws an exception.

Usage

The transaction attribute determines how methods in your component participate in transactions; at the component level, the setting affects all methods. You can also set a transaction attribute for methods within a component using the com.sybase.jaguar.method.tx_type method property. Values set at the method level override the component setting.

See also

com.sybase.jaguar.method.tx_type

com.sybase.jaguar.component.tx_vote

Description

Specifies whether the component is automatically deactivated after each method invocation, or deactivated at explicit transaction boundaries. For component types to which it applies, setting this property to true effectively marks the component as stateless.

Applies only to components that use a control interface in which the instance activation and deactivation correspond to transaction boundaries. In other words, the option does not apply to EJB components or any component that uses the control interface CtsComponents::ObjectControl.

Syntax

Allowable values are `true` and `false`. The default for new components is `true`.

Usage

In EAServer Manager, set this property with the Automatic Demarcation / Deactivation check box on the Transactions tab in the Component Properties window.

For EJB session bean components and components that use the CtsComponents::ObjectControl control interface, the com.sybase.jaguar.component.stateless property provides similar behavior.

See also

com.sybase.jaguar.component.control, com.sybase.jaguar.component.stateless

com.sybase.jaguar.component.type

Description

Specifies the component type.

Syntax

type

Where *type* is one of:

Type value	To indicate
com	ActiveX/COM
cpp	C++, using CORBA C++ language bindings
cs	C (using Open Client/Server CS datatypes)
ejb	An Enterprise Java bean (EJB) component, conforming to the EJB version indicated by the com.sybase.jaguar.component.model.version property
java	A CORBA/Java component, using CORBA Java language bindings
jdbc	A Java component using JDBC datatypes for parameters and return values
pb	A PowerBuilder component
library	A component run by the specified dispatcher library

Usage

In EAServer Manager, set this property using the Component Type field on the General tab in the Component Properties dialog box.

com.sybase.jaguar.component.XSLTfactory

Description	Specifies the class name for a custom XSLT XML parser factory class.
Syntax	The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.component.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.
See also	com.sybase.jaguar.component.DOMfactory, com.sybase.jaguar.component.SAXfactory

com.sybase.jaguar.description

Description	Specifies a text description of the component.
Syntax	*desc* Where *desc* is the descriptive text.
Usage	In EAServer Manager, set this property using the Description field on the General tab of the Component Properties dialog box.

Connection cache properties

Description	Connection cache property names begin with `com.sybase.jaguar.conncache`. In EAServer Manager, configure connection cache properties in the Connection Cache Properties dialog box. In addition to the properties described here, you can configure additional properties for JDBC and Client-Library caches. See “Advanced tab” on page 90 for more information.
See also	Chapter 4, “Database Access”

com.sybase.jaguar.conncache.cachebyname

Description	Specifies whether components can use this cache by specifying the cache name alone.
Syntax	`true` or `false`. The default is `false`.

Usage	To access a cache from an aliased JNDI reference, this property must be true. In EAServer Manager, set this property using the Allow By-Name Access check box on the Caching tab in the Connection Cache Properties dialog box.
See also	com.sybase.jaguar.component.resource-ref, com.sybase.jaguar.webapplication.resource-ref

com.sybase.jaguar.conncache.cachesize

Description	This property is obsolete and has been replaced by com.sybase.jaguar.conncache.poolsize.max. EAServer automatically converts settings that use the old name to the new name when the server is started.
Syntax	Same as com.sybase.jaguar.conncache.poolsize.max.

com.sybase.jaguar.conncache.check

Description	Specifies the query to use when testing whether a connection is still usable.
Syntax	The SQL query text. The default is: select 1 The default does not work on all databases. For example, for an Oracle database, you must set this property to: select 1 from dual
Usage	To set this property in EAServer Manager, use the Advanced tab in the Connection Cache Properties dialog box.
See also	com.sybase.jaguar.conncache.checkallowed

com.sybase.jaguar.conncache.checkallowed

Description	Specifies whether the cache manager should test connections before returning them to the cache.
Syntax	`true` or `false`. The default is `true`.
Usage	The query used to test the cache is specified by the com.sybase.jaguar.conncache.check property.

To set this property in EAServer Manager, use the Enable connection sanity check option on the Caching tab in the Connection Cache Properties dialog box.

See also com.sybase.jaguar.conncache.check

com.sybase.jaguar.conncache.cmp_stats

Description Enables statistics output for EJB CMP entity beans that use the cache.

Syntax Integer values. The default of 0 disables statistics output. A positive integer specifies how often, in seconds, the persistence engine logs statistics entries. Statistics output includes table statistics, cache usage statistics, and query statistics. This data can be useful for tuning other connection cache and component properties for best performance.

com.sybase.jaguar.conncache.config-property

Description Specifies additional driver-specific connection properties.

Syntax A comma-separated list of parenthesized quadruplets, each of which sets a driver property:

(description=*desc*, env-entry-value=*value*, env-entry-type=*type*, env-entry-name=*name*)

Where:

Variable	Specifies
desc	An optional description.
value	The property value.
type	The Java datatype of the property value.
name	The property name.

Examples **Example 1** This example sets a string property named "DYNAMIC_PREPARE" to a value of "true":

```
(description=some description, env-entry-value=true,
env-entry-type=java.lang.String,
env-entry-name=DYNAMIC_PREPARE)
```

Example 2 This example configures two properties named "prop1" and "prop2," of type Integer, to values of 1 and 2, respectively. The description of the second property has been omitted:

```
(description="Set prop1", env-entry-value=1,
env-entry-type=java.lang.Integer,
env-entry-name=prop1), (description=,
env-entry-value=2, env-entry-type=java.lang.Integer,
env-entry-name=prop2)
```

Usage

For JDBC 2.0 drivers, the properties are set using reflection, which finds a method that matches the property name and datatype. For example, if you set the property packetSize to a java.lang.Integer value, the cache manager looks for a method named setPacketSize that takes a parameter of this type. For JDBC 1.0 drivers, the properties are set by creating a java.utils.Properties instance that contains the property settings and passing it to the driver.

You can also set properties by adding them to the cache properties directly. Any property whose name does not begin with com.sybase.jaguar is passed to the driver as a String property setting. You cannot set properties of type other than String this way.

com.sybase.jaguar.conncache.conlibdll

Description

Specifies the DLL, shared library, or JDBC driver class name used in the connection.

Syntax

The syntax is the same as the DLL or Class Name field on the Driver tab in the Connection Cache Properties dialog box. See “Driver properties” on page 84.

com.sybase.jaguar.conncache.conlibname

Description

Specifies the type of connection cache.

Syntax

Your choices for library type are:

Table B-3: Library type values

Value	To indicate
CTLIB_110	Sybase Open Client Client-Library connections
ODBC	Connections using an open database connectivity driver
JDBC	Connections using Java database connectivity driver
OCI_7	Connection using OCI 7.x
OCI_8	Connections using OCI 8.x

Usage

In EAServer Manager, set this property on the Caching tab in the Connection Cache properties dialog box.

com.sybase.jaguar.conncache.db_type

Description	Specifies the database type.
Syntax	The name of an existing database type entity.
See also	Database type properties

com.sybase.jaguar.conncache.description

Description	Specifies an optional text description of the cache.
Syntax	*desc* Where *desc* is the descriptive text.
Usage	In EAServer Manager, set this property in the Description field on the General tab in the Connection Cache properties dialog box.

com.sybase.jaguar.conncache.highavailability

Description	Specifies whether to use a high availability connection.
Syntax	`true` or `false`. The default is `false`.
Usage	In EAServer Manager, set this property on the Driver tab in the Connection Cache properties dialog box. The property is available only for Client Library 11.0 connections.

com.sybase.jaguar.conncache.idletimeout

Description	Specifies the number of seconds an idle connection remains in the pool before it is dropped.
Syntax	The default is 300 seconds (5 minutes).
Usage	In EAServer Manager, set this property on the Caching tab in the Connection Cache properties dialog box.

com.sybase.jaguar.conncache.name

Description	Specifies the cache name.

Syntax	The cache name as displayed in EAServer Manager.

com.sybase.jaguar.conncache.password.e

Description	Specifies the password for connections in the cache.
Syntax	The password text. Values are encrypted in the repository.
Usage	In EAServer Manager, set this property in the Password field on the General tab in the Connection Cache Properties dialog box.
See also	com.sybase.jaguar.conncache.username

com.sybase.jaguar.conncache.poolmanager.maxconnection

Description	Specifies the maximum number of connections that can be allocated before the value of the com.sybase.jaguar.conncache.poolmanager.wait property determines whether to wait for a connection. The number of connections should be greater than or equal to the value of com.sybase.jaguar.conncache.poolsize.max. A value of 0 indicates that new connections should be opened whenever they are required. Excess connections are deallocated and not returned to the cache.
Syntax	The number of connections.
Usage	In EAServer Manager, set this property in the Maximum Connections field on the Caching tab in the Connection Cache Properties dialog box.
See also	com.sybase.jaguar.conncache.poolsize.max

com.sybase.jaguar.conncache.poolsize.max

Description	Specifies the number of connections kept open in the cache.
Syntax	A positive integer. If not set, the default is 10.
Usage	The cache size specifies the number of connections kept open in the cache. At times, more connections may be open than the size allows if additional connections are required. However, excess connections are deallocated and not returned to the cache. In EAServer Manager, set this property in the Maximum Connection Cache Pool Size field on the Caching tab in the Connection Properties dialog box.

See also	com.sybase.jaguar.conncache.poolsize.min

com.sybase.jaguar.conncache.poolsize.min

Description	Specifies the minimum number of connections kept open in the cache.
Syntax	A positive integer. If not set, the default is 0.
Usage	When the server starts, it preallocates and opens the specified number of connections. In EAServer Manager, set this property in the Minimum Connection Cache Pool Size field on the Caching tab in the Connection Properties dialog box.
See also	com.sybase.jaguar.conncache.poolsize.max

com.sybase.jaguar.conncache.refreshrate

Description	The refresh rate for the connection cache.
Syntax	The default is 600 seconds (10 minutes).
Usage	In EAServer Manager, set this property on the Caching tab in the Connection Cache properties dialog box.

com.sybase.jaguar.conncache.remotesvrname

Description	The “server name” to connect to, which may be a URL for JDBC connection caches.
Syntax	See “General properties” on page 79. The syntax is the same as for the Server Name field in the EAServer Manager Connection Cache Properties dialog box.

com.sybase.jaguar.conncache.replaceorwithcolon

Description	If this property is set to true, pipe characters, ‘|’, in the server URL for JDBC connection caches are replaced with ‘:’ before calling the JDBC driver’s setServerName() method.
Syntax	`true` or `false`. The default is `false`.

com.sybase.jaguar.conncache.ssa

Description — Enables set-proxy support for connections to databases that support this feature.

Syntax — `true` or `false`. The default is `false`, which disables set-proxy support.

Usage — Current versions of Adaptive Server Enterprise and Adaptive Server Anywhere allow a user to assume the identity and privileges of another user. You can use this feature with any database that recognizes this command:

```
set session authorization "login-name"
```

When proxy support is enabled, connections retrieved from the cache are set to act as a proxy for the user name associated with the EAServer client. To set proxy to another user name, use the Java JCMCache.getProxyConnection() method or the C JagCmGetProxyConnection() routine in your component.

The user name specified in the cache properties (com.sybase.jaguar.conncache.username) must have set-proxy privileges in the database and/or server used by the cache.

In EAServer Manager, set this property using the Advanced tab in the Connection Cache Properties dialog box.

See also — com.sybase.jaguar.conncache.username, com.sybase.jaguar.conncache.ssa.systemid

com.sybase.jaguar.conncache.ssa.systemid

Description — Specifies an alternate user name to set proxy to if the cache is used by an EJB CMP entity bean and the cache has set-proxy support enabled.

Syntax — The user name. If not set, the default is the value of com.sybase.jaguar.conncache.username.

Usage — In an EJB CMP entity bean, the client user name is not available to set proxy to since the persistence engine runs as the system user. In this case, if a user name is specified with this property, the cache manager sets proxy to the specified user. Otherwise, it sets proxy to the cache's User Name.

See also — com.sybase.jaguar.conncache.ssa, com.sybase.jaguar.conncache.username

com.sybase.jaguar.conncache.username

Description — Specifies the user name for connections in the cache.

Syntax	The user name.
Usage	In EAServer Manager, set this property using the User name field on the General tab in the Connection Cache Properties dialog box.
See also	com.sybase.jaguar.conncache.password.e

com.sybase.jaguar.conncache.wait

Description	Specifies whether to wait for a connection, when the number of connections that have been allocated is equal to the value of com.sybase.jaguar.conncache.poolmanager.maxconnection.
Syntax	`true` or `false`. The default is `true`.
Usage	In EAServer Manager, set this property in the Wait for Connection field on the Caching tab in the Connection Cache Properties dialog box.
See also	com.sybase.jaguar.conncache.poolsize.max com.sybase.jaguar.conncache.poolsize.min com.sybase.jaguar.conncache.poolmanager.maxconnection

Connector properties

Description	Connector property names begin with `com.sybase.jaguar.connector`. In EAServer Manager, configure connector properties in the Connector Properties dialog box.
See also	“Configuring connectors” on page 94

com.sybase.jaguar.connector.auth-mechanism

Description	Specifies the authentication mechanism supported by the resource adapter (connector.)
Syntax	For example: (auth-mech-type=basic-password, auth-mech-cred=javax.resource.security.PasswordCredential)
Usage	This property is defined in the deployment descriptor and set automatically when you import the connector resource archive (RAR) or JAR file.

com.sybase.jaguar.connector.config-property

Description	Defines a single configuration property for a managed connection factory.
Syntax	For example: (env-entry-value=jdbc:cloudscape:rmi:CloudscapeDB; create=true, env-entry-type=java.lang.String, env-entry-name=ConnectionURL)
Usage	This property is defined in the deployment descriptor and set automatically when you import the connector RAR or JAR file.

com.sybase.jaguar.connector.connection-impl-class

Description	Specifies the connector's implementation class.
Syntax	The class name in dot notation; for example: com.sun.connector.blackboxJdbcConnection
Usage	The class name is defined in the deployment descriptor and set automatically when you import the connector RAR or JAR file.

com.sybase.jaguar.connector.connection-interface

Description	Specifies the name of the connection interface supported by the connector.
Syntax	For example: java.sql.Connection
Usage	This property is defined in the deployment descriptor and set automatically when you import the connector RAR or JAR file.
See also	com.sybase.jaguar.connector.connectionfactory-interface com.sybase.jaguar.connector.managedconnectionfactory-class

com.sybase.jaguar.connector.connectionfactory-impl-class

Description	Specifies the connection factory class that implements the connector-specific ConnectionFactory interface.
Syntax	The class name in dot notation; for example: com.sun.connector.blackboxJdbcConnection

Usage	This property is defined in the deployment descriptor and set automatically when you import the connector RAR or JAR file.
See also	com.sybase.jaguar.connector.managedconnectionfactory-class, com.sybase.jaguar.connector.connection-interface

com.sybase.jaguar.connector.connectionfactory-interface

Description	Specifies the connection factory interface supported by the connector.
Syntax	For example: javax.sql.DataSource
Usage	This property is defined in the deployment descriptor and set automatically when you import the connector RAR or JAR file.
See also	com.sybase.jaguar.connector.managedconnectionfactory-class, com.sybase.jaguar.connector.connectionfactory-impl-class

com.sybase.jaguar.connector.display-name

Description	The name that identifies this connector in EAServer Manager.
Syntax	connector-name For example: BlackBoxLocalTx

com.sybase.jaguar.connector.eis-type

Description	Specifies the Enterprise Information System (EIS) type to which connections are made.
Syntax	For example, “JDBC Database”
Usage	The EIS type is defined in the deployment descriptor and set automatically when you import the connector RAR or JAR file.

com.sybase.jaguar.connector.enabled

Description	Specifies whether the connector is enabled or disabled.

Syntax	`true` or `false`. The default is `true`.
Usage	In EAServer Manager, set this property on the Advanced tab of the Connector Properties dialog box.

com.sybase.jaguar.connector.icon.large-icon

Description	Specifies the name of the large icon file associated with the connector.
Syntax	A *.gif* or *.jpg* file name.

com.sybase.jaguar.connector.icon.small-icon

Description	Specifies the name of the small icon file associated with the connector.
Syntax	A *.gif* or *.jpg* file name.

com.sybase.jaguar.connector.idletimeout

Description	Specifies the number of seconds an idle connection remains in the pool before it is dropped.
Syntax	*n* Where *n* is a positive integer.
Usage	In EAServer Manager, set the Idle Connection Timeout on the General tab in the Connector Properties dialog box.

com.sybase.jaguar.connector.java.classes

Description	Specifies additional classes to be reloaded when the connector is refreshed.
Syntax	Same as for com.sybase.jaguar.application.java.classes.
Usage	In EAServer Manager, specify the Java Class file names on the Java Classes tab in the Connector Properties dialog box.
See also	com.sybase.jaguar.component.java.classes

com.sybase.jaguar.connector.managedconnectionfactory-class

Description	Specifies the Java class that implements the javax.resource.spi.ManagedConnectionFactory interface.
Syntax	The class name in dot notation; for example: com.sun.connector.blackbox.LocalTxManagedConnectionFactory
Usage	This property is set automatically when you import the connector RAR or JAR file.

com.sybase.jaguar.connector.name

Description	The name that identifies this connector in the repository.
Syntax	connector-name
Usage	The connector name is set automatically when you import the connector RAR or JAR file.

com.sybase.jaguar.connector.queuesize

Description	Specifies the connector's connection pool size.
Syntax	*n* Where *n* is a positive integer; the default is 10.
Usage	In EAServer Manager, set the Configured Queue Size on the General tab in the Connector Properties dialog box. When a component asks for a connection, EAServer attempts to get a connection from the pool. If none exists, it opens a new connection.
See also	"Configuring a connector" on page 95

com.sybase.jaguar.connector.reauthenticationsupport

Description	Specifies whether the connector supports reauthentication of an existing managed connection factory instance.
Syntax	`true` or `false`. The default is `false`.
Usage	This property is set automatically when you import the connector RAR or JAR file.

com.sybase.jaguar.connector.security-permission

Description — Specifies a security permission that is required by the connector.

Syntax — *security-permission*

Where *security-permission* is other than those required by the default permission set defined in the connector specification.

Usage — The security permission property is set automatically when you import the connector RAR or JAR file.

com.sybase.jaguar.connector.spec-version

Description — Specifies the connector architecture specification version number that is supported by the connector.

Syntax — *version*

Where *version* is the specification version number.

Usage — The specification version number is set automatically when you import the connector RAR or JAR file.

See also — com.sybase.jaguar.connector.version

com.sybase.jaguar.connector.transaction-support

Description — Specifies the level of transaction support provided by the connector.

Syntax — The transaction type can be one of:

Transaction type	Description
NO_TRANSACTION	The connector is nontransactional; it does not support local transactions or XA resources.
LOCAL_TRANSACTION	The connector implements only the LocalTransaction interface, therefore, EAServer must manage all transactions.
XA_TRANSACTION	The connector supports both local transactions and XA resources.

Usage — This property is set automatically when you import the connector RAR or JAR file.

See also — “Transaction modes” on page 94.

com.sybase.jaguar.connector.vendor-name

Description	Specifies the connector's vendor name.
Syntax	*vendor* Where *vendor* is the name of the vendor.
Usage	The vendor-name property is set automatically when you import the connector RAR or JAR file.

com.sybase.jaguar.connector.version

Description	Specifies the version number of the connector software.
Syntax	*version* Where *version* is the version number.
Usage	The connector version is set automatically when you import the connector RAR or JAR file.
See also	com.sybase.jaguar.connector.spec-version

Database type properties

Description

In connection caches used to support automatic persistence, stateful failover, or the EAServer message service, you must configure the Database Type connection cache property. This property defines database-specific information required by the storage component, for example, the commands to verify that a table exists and create new tables. Several types are predefined, as described in Table 4-6 on page 91.

You can create your own database type definitions by copying one of the existing files in the EAServer *Repository/DatabaseType* directory, then editing the property settings. Any changes you make to the predefined database type files may be overwritten by subsequent EAServer installs.

jagtool and the Repository API do not support manipulation of database type properties. To deploy a custom database type with your application, include the database type file in the .files property of the entity that you are deploying.

The database type properties include the following:

- Logical column type definitions, which map types used in EJB-CMP field mappings to database column types. These properties use the logical type name as the property name.
- Configuration properties, prefixed with com.sybase.jaguar, as in other entity configurations.
- Optional ejbQuery properties, which can be included if the JDBC driver and database do not implement EJB-QL functions.

See also com.sybase.jaguar.conncache.db_type,
"Database type setting" on page 91

Logical column type definitions

Description Type definitions specify the format to define a column of the specified type in a SQL/DDL create table statement for the target database. These types can be used in the EJB CMP field mapping properties rather than the actual database types. EAServer uses the specified syntax to create columns for each container managed field in the bean.

Syntax Use this syntax to define a column type in the database type properties:

typeName=columnSpec

Where *typeName* is the logical type name used in field mapping properties, and *columnSpec* is the database syntax to define a column of the type in a create table statement. Table B-4 describes the logical type names used in the predefined EAServer database types. In component field mapping properties, using the logical type names rather than actual database types allows you to more easily run the same configuration against databases of different types.

Table B-4: Logical type names

Name	Represents
`byte`	8-bit integer.
`short`	16-bit integer.
`int`	32-bit integer.
`long`	64-bit integer.
`float`	32-bit floating point.
`double`	64-bit floating point.
`integer`	Unlimited precision integer.
`integer(`*`precision`*`)`	Integer with the specified precision.
`decimal`	Decimal.

Name	Represents
`decimal(`*`precision`*`,`*`scale`*`)`	Decimal with the specified precision and scale.
`boolean`	Boolean (bit).
`char`	A single character.
`string`	Variable length character data.
`string(`*`length`*`)`	Variable length character data, with the specified maximum value length.
`string(`*`length`*`) fixed length`	Fixed length character data, with the specified maximum value length.
`binary`	Variable length binary data.
`binary(`*`length`*`)`	Variable length binary data, with the specified maximum value length.
`binary(`*`length`*`) fixed length`	Fixed length binary data, with the specified maximum value length.
`date`	Date values.
`time`	Time values.
`datetime`	Date plus time values.
`dbts`	The database timestamp type. Map this logical type to the database timestamp type by setting the com.sybase.jaguar.databasetype.dbts property.

In the type name, you can use named placeholders for length, precision, or scale specifiers. For example, for variable-length character data, you can use `string(maxLength)` where *maxLength* represents the field length specified in the field mapping property that uses the type. For decimal or money data, you might use `decimal(precision,scale)`. You can use the placeholder names in the column specification, as shown in the examples provided below.

columnSpec can include variable expressions to be evaluated at runtime, using this syntax:

${*expression*]

The expression language is that used for C language preprocessor #if and #ifdef directives. You can embed the name of a placeholder for the length, precision, or scale specifier used in the type name, as well as the predefined placeholders described in Table B-5.

Table B-5: Predefined placeholders for column type definitions

Placeholder	To indicate
`column`	The column name specified in field mapping properties.
`dbid`	A Boolean that evaluates to true if the database supports identity types or the equivalent; that is, the property com.sybase.jaguar.databasetype.dbid is set to true.
`isNull`	Evaluates to the string "null" if the field mapping allows null values and to "not null" otherwise.
`notNull`	Boolean that is true when `isNull` evaluates to "not null".

Examples

Simple character type example This example defines a variable length character type, for a database that uses varchar for this column type, and has no database-imposed limit on the maximum length of varchar columns:

```
string(maxLength)=${column} varchar(${maxLength}) ${isNull}
```

If a field mapping uses the value `lastName[string(100) not null]`, EAServer creates the mapped column as:

```
lastName varchar(100) not null
```

In this instance, the *column* placeholder is replaced with the column name, `lastName`, *maxLength* is replaced by the declared length, 100, and *isnull* is replaced by `not null`.

Character data that uses different database column types depending on length This example defines a variable length character type, for a database that uses type varchar for columns to 255 characters wide, but requires type text for wider columns. This example is copied from *Sybase_ASE.props*:

```
string(maxLength)=${column} ${maxLength > 255 ? "text" : ("varchar(" +
Number.toString(maxLength) + ")")} ${isNull}
```

The example uses the ? operator to declare the column as varchar or text, depending on whether the declared length is greater than 255.

See also

"Setting field-mapping properties" in the *EAServer Programmer's Guide*.

com.sybase.jaguar.databasetype.afterInsert

Description

Used when inserting rows for a CMP entity bean when auto-key generation is enabled. This property specifies the database syntax to select the value of a generated key after a row is inserted.

Syntax

The text of a SQL select statement that returns a single-column, single-row result set containing the generated key value.

Usage

Syntax to specify commands for the afterInsert, checkDelete, and checkUpdate properties For these properties that specify database command text, you can use the placeholders in the following table:

Placeholder	Specifies
`${`*`expression`*`}`	A text expression in the language used for C language preprocessor #if and #ifdef directives.
`dbid`	Useful for Sybase databases and those that use similar native syntax to generate key values, such as Microsoft SQL Server. When used in expressions, evaluates to true if auto key generation is enabled and keys are generated using Sybase identity column or the equivalent.
`sequence`	Useful for Oracle databases. When used in expressions, the sequence name specified by the component property com.sybase.jaguar.component.db.sequence. Can also be used when using a key generation table, in which case it evaluates to the key table name plus column name.
`generateKey`	When used in expressions, a boolean that is true if automatic key generation is enabled for the component (com.sybase.jaguar.component.generateKey is true).
`table`	When used in expressions, the table name specified in the component property com.sybase.jaguar.component.storage.
`timestamp`	When used in expressions, the component's timestamp column specified by component property com.sybase.jaguar.component.timestamp

If the evaluated text begins with '+' (plus sign), the text after '+' is appended to the same batch as the insert command, with '+' replaced by a space. If the evaluated text begins with ';' (semicolon), the entire text is appended as-is to the same batch as the insert command, including the semicolon. Otherwise, the text is sent to the database in a separate batch, after processing the results of the insert command.

com.sybase.jaguar.databasetype.checkDelete

Description — Specifies the SQL text to verify that a delete affected one row. Used when the component uses optimistic concurrency control, and the isolation level requires update verification.

Syntax — The text of the database command to verify that the delete affected one and only one row. If the delete affected more or less rows, the command text must raise an error, for example, using the SQL print or raiserror command. The text can be specified to run in the same batch as the update, or in a new batch, as described in "Syntax to specify commands for the afterInsert, checkDelete, and checkUpdate properties" on page 436.

For example, if using Sybase Adaptive Server Enterprise:

```
+ if @@rowcount <> 1 print 'OCC: Delete Failed: ${table}'
```

Usage — There are two mechanisms that you can configure to verify updates. Use the one that works best for your database and JDBC driver:

- Set the properties com.sybase.jaguar.databasetype.checkDelete and com.sybase.jaguar.databasetype.checkUpdate to specify the commands required to verify updates or deletes, respectively.
- Set the com.sybase.jaguar.databasetype.checkUpdateCount property to true, so that the CMP engine verifies updates and deletes by checking the row count returned to the JDBC driver.

com.sybase.jaguar.databasetype.checkUpdate

Description — Specifies the SQL text to verify that an update affected one row. Used when the component uses optimistic concurrency control, and the isolation level requires update verification.

Syntax — The text of the database command to verify that the update affected one and only one row. If the update affected more or less rows, the command text must raise an error, for example, using the SQL print or raiserror command. The text can be specified to run in the same batch as the update, or in a new batch, as described in "Syntax to specify commands for the afterInsert, checkDelete, and checkUpdate properties" on page 436. For example, if using Sybase Adaptive Server Enterprise:

```
+ if @@rowcount <> 1 print 'OCC: Update Failed: ${table}'
```

Usage — The Usage section for com.sybase.jaguar.databasetype.checkDelete describes how to configure the mechanism that performs update verification.

com.sybase.jaguar.databasetype.checkUpdateCount

Description — Specifies whether the CMP engine should check the row count returned from the JDBC driver to verify that an update or delete affected one and only one row.

Syntax — `true` or `false`.

Usage — The Usage for com.sybase.jaguar.databasetype.checkDelete describes how to configure the mechanism that performs update verification.

com.sybase.jaguar.databasetype.columnAlias

Description — Specifies the syntax to generate compact column names, which can improve performance by reducing the size of query result sets.

Syntax — Specify the syntax to rename a column in the query's column list. For example, if you specify `_[index]=[column,` queries use column lists of the form `select _1=column1, _2=column2, ...`

If you specify `[column] as C[index]`, queries use column lists of the form `select col1 as C1, col2 as C2, ....`

Specify the syntax that is legal for your database server.

com.sybase.jaguar.databasetype.createSequence

Description — Used in the following scenarios:

- For Oracle and similar databases that generate key values using a named sequence, specifies the command text to create the sequence.
- For any database, specifies the syntax to create a table that is used to generate key values when the component properties require automatic key generation.

Syntax — The command text to create the sequence. You can use the placeholders described in the following table:

Placeholder	Specifies
`${`*`expression`*`}`	A text expression in the language used for C language preprocessor #if and #ifdef directives.

Placeholder	Specifies
`sequence`	When used in expressions, the Oracle sequence name or key table name specified by the component property com.sybase.jaguar.component.db.sequence.
`next_key`	When using a key table, the column name that contains the next key value, as specified by the component property com.sybase.jaguar.component.db.sequence.

For example, if using Oracle:

```
create sequence ${sequence}
```

See also

com.sybase.jaguar.databasetype.generateKey, com.sybase.jaguar.component.db.sequence

"Enabling automatic key generation" in Chapter 27, "Creating Entity Components," in the *EAServer Programmer's Guide*

com.sybase.jaguar.databasetype.dbid

Description

Specifies whether the database supports auto key generation using the Sybase identity column type or the equivalent.

Syntax

`true` or `false`.

com.sybase.jaguar.databasetype.dbts

Description

If the database generates row timestamps, specifies the type of the timestamp column.

Syntax

typeName[, *convertFunc*]

Where

- *typeName* is the type of the timestamp column, as defined in the properties file. See Logical column type definitions on page 433 for more information.
- *convertFunc* is the database conversion function to be used when selecting the timestamp column in queries. Use the `${dbts}` placeholder to indicate the placement of the timestamp column name in the text. If you specify a conversion function, *typeName* should match the result after conversion.

Usage — If the timestamp column can be selected directly, the datatype of the timestamp column. If the timestamp column requires conversion, specify the conversion function syntax after the datatype, separated by a comma. For example, the following is the definition in *Sybase_ASA.props*:

```
string(26),dateformat(${dbts},'yyyy-mm-dd hh:nn:ss.ssssss')
```

See also — Logical column type definitions

com.sybase.jaguar.databasetype.deadlock

Description — Specifies error text to search for in a SQLException to determine whether an error indicates a database deadlock condition.

Syntax — The text to search for. The search algorithm is case-insensitive and requires an exact match. If not set, the default is `<unknown>`.

com.sybase.jaguar.databasetype.duplicateKey

Description — Specifies error text to search for in a SQLException to determine whether an error indicates an attempt to insert duplicate keys.

Syntax — The text to search for. The search algorithm requires an exact match but is not case sensitive. If not set, the default is `<unknown>`.

com.sybase.jaguar.databasetype.emptyBinary

Description — Specifies whether the database and JDBC driver can store zero-length binary values.

Syntax — `true` or `false`. If not set, the default is `true`. If set to `false`, zero-length binary values are converted to a value of length one, containing a null byte.

Usage — Some databases, such as Sybase Adaptive Server Enterprise and Microsoft SQL Server, cannot store a zero-length binary value.

See also — com.sybase.jaguar.databasetype.emptyString

com.sybase.jaguar.databasetype.emptyString

Description — Specifies whether the database and JDBC driver can store zero-length variable-length character values.

Syntax — `true` or `false`. If not set, the default is `true`. If set to `false`, zero-length values are converted to a single space.

Usage — Some databases, such as Sybase Adaptive Server Enterprise and Microsoft SQL Server, cannot store a zero-length character value.

See also — com.sybase.jaguar.databasetype.emptyBinary

com.sybase.jaguar.databasetype.generateKey

Description — Specifies the command text to specify an automatically generated key value in a database insert command.

Syntax — The command text. You can use the placeholders described in the following table:

Placeholder	Specifies
`${`*`expression`*`}`	A text expression in the language used for C language preprocessor #if and #ifdef directives.
`sequence`	When used in expressions, the Oracle sequence name or key table name specified by the component property com.sybase.jaguar.component.db.sequence.

For example, the following text selects the next value from an Oracle sequence:

```
${sequence}.nextval
```

See also — com.sybase.jaguar.databasetype.createSequence

com.sybase.jaguar.databasetype.jdbc.setBytes.maxLength

Description — Specifies the maximum length for values passed to the JDBC driver implementation of the PreparedStatement.setBytes method.

Syntax — The maximum length.

Usage — Some JDBC drivers cannot handle large values in the PreparedStatement.setBytes method. If this property is set, values greater than the maximum are passed as follows:

- For databases besides Oracle, values are passed using the PreparedStatement.setBinaryStream.
- For Oracle databases, the value is passed using special code that is specific to the Oracle JDBC driver. To ensure that large character values are passed correctly, verify that the CMP field mappings map to BLOB or an equivalent logical type if the field can hold large values. That is, the field is a binary (Java byte[] or Object) field with no length restriction, or a maximum length greater than supported by the JDBC driver's PreparedStatement.setBytes implementation.

See also com.sybase.jaguar.databasetype.jdbc.setString.maxLength

com.sybase.jaguar.databasetype.jdbc.setObject.tsTimeBase

Description Some databases, such as Sybase Adaptive Server Enterprise, store date and time values in the same column type. This property defines a time base for conversion between java.sql.Time values and database date values.

Syntax The time base.

com.sybase.jaguar.databasetype.jdbc.setString.maxLength

Description Specifies the maximum length for values passed to the JDBC driver implementation of the PreparedStatement.setString method.

Syntax The maximum length.

Usage Some JDBC drivers cannot handle large values in the PreparedStatement.setString method. If this property is set, values greater than the maximum are passed as follows:

- For databases besides Oracle, the value is passed using the PreparedStatement.seCharacterStream.
- For Oracle databases, the value is passed using special code that is specific to the Oracle JDBC driver. To ensure that large character values are passed correctly, verify that the CMP field mappings map to CLOB or an equivalent logical type if the field can hold large values. That is, the field is a string field with no length restriction, or a maximum length greater than that supported by the JDBC driver's PreparedStatement.setString implementation.

See also com.sybase.jaguar.databasetype.jdbc.setBytes.maxLength

com.sybase.jaguar.databasetype.jdbc.types.BIGINT

Description	Allows you to redefine the integer constant that represents a BIGINT value to the JDBC driver, instead of the value of java.sql.Types.BIGINT.
Syntax	The integer type constant. If not set, the default is the value of java.sql.Types.BIGINT. Some drivers do not support this type; in this case, set the property to specify an alternate type that supports the BIGINT precision, such as java.sql.Types.DECIMAL.

com.sybase.jaguar.databasetype.name

Description	Specifies the entity name for this database type.
Syntax	The name, which must match the base name of the properties file. For example, if the file is *MySybase_ASE.props*, set this property to `MySybase_ASE`.

com.sybase.jaguar.databasetype.oracleTriggers

Description	Specifies whether to use Oracle syntax when creating triggers.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.databasetype.sybaseTriggers

com.sybase.jaguar.databasetype.selectForUpdate

Description	Specifies database syntax to select rows with an exclusive lock.
Syntax	A select statement template, for example (from *Oracle9i.props*):

```
select [columns] [from-clause] [where-clause] for update
```

You can use the placeholders described in Table B-6.

Table B-6: Select statement template placeholders

Placeholder	Represents
`[table]`	The table name
`[touch-column]`	Columns affected by updates
`[from-clause]`	The from clause
`[where-clause]`	The where clause
`[columns]`	The column list returned by the query

If the database does not support exclusive locks, set the property to an empty value.

See also com.sybase.jaguar.databasetype.selectWithLock, com.sybase.jaguar.component.selectForUpdate, com.sybase.jaguar.component.selectWithLock

com.sybase.jaguar.databasetype.selectWithLock

Description Specifies database syntax to select rows with a shared lock.

Syntax A select statement template, for example (from *Sybase_ASE.props*):

```
select [columns] [from-clause] holdlock [where-clause]
```

You can use the placeholders described in Table B-6 on page 444.

If the database does not support shared locks, set the property to an empty value.

Usage If neither com.sybase.jaguar.databasetype.selectForUpdate and com.sybase.jaguar.databasetype.selectWithLock are set, the persistence engine uses optimistic concurrency control when the component properties request locking.

See also com.sybase.jaguar.databasetype.selectForUpdate, com.sybase.jaguar.component.selectForUpdate, com.sybase.jaguar.component.selectWithLock

com.sybase.jaguar.databasetype.statementCache

Description This property is obsolete.

Syntax `true` or `false`.

Usage	This property was used in EAServer 4.x versions and is obsolete beginning in EAServer 5.0.

com.sybase.jaguar.databasetype.sybaseTriggers

Description	Specifies whether to use Sybase syntax when creating triggers.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.databasetype.oracleTriggers

com.sybase.jaguar.databasetype.trimString

Description	Specifies the persistence engine trims trailing spaces from variable-length character values.
Syntax	`true` or `false`. The default is `false`.

com.sybase.jaguar.databasetype.verify

Description	When using optimistic concurrency control with a component isolation level less than `repeatable_read`, specifies the query syntax to verify that no change has occurred to the row that was read into the container managed fields.
Syntax	A select statement template that selects a count of rows that match the where clause, for example (from *Sybase_ASE.props*): `select count(*) [from-clause] [where-clause]` A count of 1 passes verification. A count of 0 fails verification. You can use the placeholders described in Table B-6 on page 444.
See also	com.sybase.jaguar.databasetype.verifyWithLock

com.sybase.jaguar.databasetype.verifyWithLock

Description	When using optimistic concurrency control with a component isolation level of `repeatable_read` or higher, specifies the query syntax to verify that no change has occurred to the row that was read into the container managed fields.
Syntax	A select statement template that selects a count of rows that match the where clause, for example (from *Sybase_ASE.props*):

```
select count(*) [from-clause] holdlock [where-clause]
```

A count of 1 passes verification. A count of 0 fails verification. You can use the placeholders described in Table B-6 on page 444.

If the database cannot support verification with locking, set the property to an empty value.

See also com.sybase.jaguar.databasetype.verify

com.sybase.jaguar.description

Description An optional description of the database type entity.

Syntax The text of the description.

ejbQuery properties

Description Allow you to map EJB-QL functions to syntax appropriate for the database. Most JDBC 2.0-compliant drivers support prepared statement function escape syntax, and these properties are not required. If needed, you can set additional properties to specify the database syntax for functions that the driver does not support.

Syntax ejbQuery.*FUNCNAME*(*arguments*)=*dbSyntax*

Where:

- *FUNCNAME* is the EJB-QL function name.
- *arguments* is a list of up to three argument names, separated by commas.
- *dbSyntax* is the equivalent database function. You can use the argument names as placeholders, as shown in the following example from *Sybase_ASE.props*:

```
ejbQuery.LOCATE(string1,string2)=CHARINDEX(${string1},${string2})
```

If you do not specify a mapping for a function, the EJB-QL syntax is used in database queries.

EJB local reference properties

Description

EJB components and Web applications allow you to define EJB local references that specify aliased JNDI names used in code to look up EJB components that support local invocations. These aliases allow the code to run on servers where the referenced components use a different JNDI name.

EJB reference syntax

EJB local references have the form:

(description=*desc*,ejb-link=,jaguar-link=*jndi-name*,
local-home=*local-home*,ejb-ref-type=*type*,
ejb-ref-name=*alias*,local=*local*)

Where:

- *desc* is an optional comment describing how the reference is used.
- *jndi-name* is the JNDI name of the referenced component, specified as the component's com.sybase.jaguar.component.bind.naming property.
- *local-home* is the Java local home interface name.
- *type* is "Session" for session beans and "Entity" for entity beans.
- *alias* is the aliased JNDI name used in the code. Enter the part of the JNDI name that begins with `ejb/`. For example, if your code refers to *java:comp/env/ejb/AcctBean*, enter `ejb/AcctBean`.
- *local* is the Java local interface name.

Usage

Use an EJB local reference property to invoke local beans through their local interfaces. Use an EJB reference property to invoke beans (local or remote) through their remote interfaces.

See also

EJB reference properties on page 447,
com.sybase.jaguar.component.ejb-local-ref,
com.sybase.jaguar.webapplication.ejb-local-ref

EJB reference properties

Description

Application clients, EJB components, and Web applications allow you to define EJB references that specify aliased JNDI names used in the application, component, or servlet code. These aliases allow the code to run on servers where the referenced components use a different JNDI name.

EJB reference syntax

EJB references have the form:

(description=*desc*,ejb-link=,jaguar-link=*jndi-name*,home=*home*, ejb-ref-type=*type*,ejb-ref-name=*alias*,remote=*remote*)

Where:

- *desc* is an optional comment describing how the reference is used.
- *jndi-name* is the JNDI name of the referenced component, specified as the component's com.sybase.jaguar.component.bind.naming property.
- *home* is the Java home interface name.
- *type* is "Session" for session beans and "Entity" for entity beans.
- *alias* is the aliased JNDI name used in the code. Enter the part of the JNDI name that begins with `ejb/`. For example, if your code refers to *java:comp/env/ejb/AcctBean*, enter `ejb/AcctBean`.
- *remote* is the Java remote interface name.

Usage

Use an EJB reference property to invoke beans (local or remote) through their remote interfaces. Use an EJB local reference property to invoke local beans through their local interfaces.

See also

EJB reference properties on page 447,
com.sybase.jaguar.applicationclient.ejb-ref,
com.sybase.jaguar.component.ejb-ref,
com.sybase.jaguar.webapplication.ejb-ref

Entity collection properties

Description

Entity collection properties begin with `com.sybase.jaguar.entitycollection`.

Usage

An entity collection allows you to group related entities together for easy replication to another EAServer installation via archives or synchronization.

When you install an entity in a collection, all child entities are automatically installed. For example, if you install a server, all applications, servlets, Web applications, and packages installed in that server become part of the entity collection.

EAServer Manager does not support entity collections, but you can manipulate them using jagtool or the Jaguar::Repository interface.

The *Repository/EntityCollection* directory of your EAServer installation contains a sample entity collection definition.

See also — Chapter 12, “Using jagtool and jagant”
“Deploying other entity types” on page 196

com.sybase.jaguar.description

Description — Specifies a text description of the entity collection.

Syntax — *desc*

Where *desc* is the descriptive text.

com.sybase.jaguar.entitycollection.applications

Description — Specifies application entities installed in this collection.

Syntax — A comma-separated list of application names.

See also — Application properties on page 345

com.sybase.jaguar.entitycollection.clusters

Description — Specifies cluster entities installed in this collection.

Syntax — A comma-separated list of cluster names.

See also — Cluster properties on page 354

com.sybase.jaguar.entitycollection.connectioncaches

Description — Specifies connection caches that are installed in this collection.

Syntax — A comma-separated list of connection cache names.

See also — Connection cache properties on page 418

com.sybase.jaguar.entitycollection.files

Description — Specifies additional files to be included in the archive for this entity collection. The default file set are those files returned by calling Jaguar::Repository::files() on each entity installed in the collection.

Syntax	Same as for com.sybase.jaguar.applicationclient.files.
See also	com.sybase.jaguar.component.files

com.sybase.jaguar.entitycollection.identities

Description	Specifies identities installed in this collection.
Syntax	A comma-separated list of identity names (an identity is a security entity of type `identity`).
See also	Security properties on page 500

com.sybase.jaguar.entitycollection.listeners

Description	Specifies listeners that are installed in this collection.
Syntax	A comma-separated list of listener names.
See also	Listener properties on page 456

com.sybase.jaguar.entitycollection.name

Description	Specifies the name of this collection.
Syntax	*name* Where *name* is the collection name.

com.sybase.jaguar.entitycollection.packages

Description	Specifies packages that are installed in this collection.
Syntax	A comma-separated list of package names.
See also	Package properties on page 483

com.sybase.jaguar.entitycollection.roles

Description	Specifies roles that are installed in this collection.
Syntax	A comma-separated list of role names.

See also	Role properties on page 497

com.sybase.jaguar.entitycollection.securityprofiles

Description	Specifies security profiles installed in this collection.
Syntax	A comma-separated list of security profile names (a security profile is a security entity of type `listener`).
See also	Security properties on page 500

com.sybase.jaguar.entitycollection.servers

Description	Specifies servers that are installed in this collection.
Syntax	A comma-separated list of server names.
See also	Server properties on page 505

com.sybase.jaguar.entitycollection.servlets

Description	Specifies servlets that are installed in this collection.
Syntax	A comma-separated list of servlet names.
See also	Servlet properties on page 563

com.sybase.jaguar.entitycollection.webapplications

Description	Specifies Web applications that are installed in this collection.
Syntax	A comma-separated list of Web application names.
See also	Web application properties on page 575

Environment properties

Description

Application clients, EJB components, and Web applications allow you to define environment properties, which contain global read-only data used by the application code. For example, you might define the administrator's e-mail address as an environment property.

Environment property syntax

Environment properties have the form:

env-entry1, *env-entry2*, ...

Where *env-entry1*, *env-entry2*, and so forth are of the form:

(description=*desc*,env-entry-value=*value*,
env-entry-type=*type*,env-entry-name=*jndi-name*)

Where:

- *desc* is an optional comment describing how the entry is to be set.
- *value* is the initial value of the property, specified in a format that is valid for the selected datatype.
- *type* is the Java datatype of the entry, which is one of:
  - java.lang.String
  - java.lang.Boolean
  - java.lang.Integer
  - java.lang.Double
  - java.lang.Byte
  - java.lang.Short
  - java.lang.Long
  - java.lang.Float
- *jndi-name* is entry's JNDI name, relative to the *java:comp/env* prefix.

See also

com.sybase.jaguar.applicationclient.env-entry,
com.sybase.jaguar.component.env-entry,
com.sybase.jaguar.webapplication.env-entry

Filter properties

Description — Filter properties begin with `com.sybase.jaguar.filter`. A filter is a Java class that can be used in a Web application to filter the HTTP response returned to Web clients. For information on implementing filters, see Chapter 23, "Using Filters and Event Listeners," in the *EAServer Programmer's Guide*.

See also — Web application properties on page 575, Servlet properties on page 563

com.sybase.jaguar.filter.class

Description — The Java class that implements the filter.

Syntax — The class name, for example:

```
com.foo.YourResponseFilter
```

com.sybase.jaguar.filter.description

Description — An optional text description of the filter.

Syntax — The descriptive text.

com.sybase.jaguar.filter.init-param

Description — Initialization parameters for the filter.

Syntax — A string of the form:

entry1, *entry2*, ...

Where *entry1*, *entry2*, and so forth are of the form:

(description=*desc*,value=*value*,name=*name*)

Where:

- *desc* is an optional comment describing how the parameter is to be set.
- *value* is the value of the parameter.
- *name* is the parameter's name.

See also — Chapter 23, "Using Filters and Event Listeners," in the *EAServer Programmer's Guide*

com.sybase.jaguar.filter.large-icon

Description — Specifies the name of the large icon file associated with the filter. This property is not used in EAServer, but accommodated to comply with the Servlet 2.3 Web archive descriptor.

Syntax — A file name.

com.sybase.jaguar.filter.name

Description — The name that identifies this filter in the repository.

Syntax — *web-app*/*filter-name*

Where:

- *web-app* is the name of the Web application containing the filter.
- *filter-name* is the filter name, as displayed in EAServer Manager.

For example:

```
WebTier/MyFilter
```

com.sybase.jaguar.filter.small-icon

Description — Specifies the name of the small icon file associated with the filter. This property is not used in EAServer, but accommodated to comply with the Servlet 2.3 Web archive descriptor.

Syntax — A file name.

Instance pool properties

Description — Instance pooling allows component instances to be reused, avoiding the performance overhead of creating new instances for each client session. Each component that runs in EAServer is assigned to an instance pool, which is specified by the com.sybase.jaguar.component.instancePool property.

Usage — These instance pools are preconfigured:

Table B-7: Preconfigured instance pools

Name	Description	Maximum number of instances
SystemPool	The default for components whose com.sybase.jaguar.component.bind.thread property is set to false.	infinite, denoted by -1
BindThread	The default for components whose com.sybase.jaguar.component.bind.thread property is set to true.	512

You can create new instance pools using EAServer Manager. By default, the maximum number of instances is set to 512. Sybase recommends that you do not create an instance pool with the maximum number of instances set to `-1` (infinite). If you need an instance pool with an unlimited number of instances, use the preconfigured SystemPool.

See also com.sybase.jaguar.component.instancePool, com.sybase.jaguar.component.bind.thread

com.sybase.jaguar.instancepool.debug

Description Enables and disables logging of activity for this pool.

Syntax Specify `true` to enable debugging. The default of `false` disables debugging.

Usage Debug information is written to the server log file. You must refresh the pool or restart the server for the change to take effect.

com.sybase.jaguar.instancepool.maximum

Description Specifies the maximum number of instances that can be placed in the pool.

Syntax A decimal integer. The default is 512.

com.sybase.jaguar.instancepool.name

Description The name of the instance pool.

Syntax Enter the pool name.

JMS entity properties

Description — To use the message service, you need to add and configure the properties for these message service components:

- Message queues
- Message topics
- Queue connection factories
- Thread pools
- Topic connection factories

See also — To configure properties for these components, see:

- For message queues and message topics, “Permanent destinations” on page 165
- For queue connection factories and topic connection factories, “Connection factories” on page 168
- For thread pools, “Thread pools” on page 174

Listener properties

Description — Listener property names begin with `com.sybase.jaguar.listener`. Listeners define the protocols, port numbers, and optional SSL security profiles for client connections to a server.

See also — Server properties on page 505

com.sybase.jaguar.listener.host

Description — Specifies the host name or IP address for the listener.

Syntax — The host name or IP address, or one of the following placeholders:

Placeholder	To indicate
`${JAGUAR_HOST_NAME}`	The name of the host where the server process is running.
`${JAGUAR_IP_ADDRESS}`	The IP address of the host where the server is running.

Use one of the placeholders if you must use the same listener configuration in multiple EAServer installations. Include the domain if your server accepts internet connections, otherwise clients outside your domain cannot connect.

If your machine supports multiple network addresses, you can enter each host name or IP address in a list separated by commas. You can also enter the special value `0.0.0.0`, which causes EAServer to listen on all of the machine's host or IP addresses. When using multiple host addresses, EAServer creates a listener for each host on the specified port.

com.sybase.jaguar.listener.http.conn.keepalive

Description	Specifies the time in seconds to keep open the server side of an idle HTTP connection.
Syntax	The time in seconds. If not specified, the default is 120.

com.sybase.jaguar.listener.http.conn.maxrequests

Description	Specifies the maximum number of HTTP requests to service before closing each connection.
Syntax	The maximum, as a positive integer. If not specified, the default is 100.

com.sybase.jaguar.listener.http.connector_events

Description	Must be set to true if you use the EAServer Web server redirector between HTTP clients and the EAServer HTTP listener.
Syntax	`true` or `false`. The default is `false`.

Warning! This property must be set to true when clients connect via the Web server redirector. Otherwise, the HTTP response can be incorrect.

com.sybase.jaguar.listener.logsslerr

Description	Specifies whether to log additional SSL error information for connections handled by the listener.

Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.security.logsslerr

com.sybase.jaguar.listener.monitor.MaxRespTime

Description	Specifies the maximum allowable response time for each request. If the average response time rises above this limit, EAServer blocks additional connections until the average drops below the specified limit.
Syntax	The maximum allowable response time, in seconds. The default is -1, which indicates no time limit.
See also	com.sybase.jaguar.listener.monitor.MinInstance Chapter 9, "Using the Performance Monitor," in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.listener.monitor.MinInstance

Description	When response time monitoring is in effect (com.sybase.jaguar.listener.monitor.MaxRespTime is set to a non-negative value), specifies the minimum numer of clients that must be allowed to execute regardless of observed response times.
Syntax	The minimum number of clients. The default is -1, which means no new connections are blocked by the Performance Monitor.
See also	com.sybase.jaguar.listener.monitor.MaxRespTime Chapter 9, "Using the Performance Monitor," in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.listener.name

Description	Specifies the listener name.
Syntax	*Server_listener* Where *Server* is the server name where this listener is installed, and *listener* is the unqualified listener name. For example: Jaguar_iiops1

com.sybase.jaguar.listener.port

Description: Specifies the port number.

Syntax: The integer port number.

com.sybase.jaguar.listener.protocol

Description: Specifies the protocol for connections accepted by this listener.

Syntax: Allowable values are:

- HTTP
- IIOP
- TDS
- HTTPS
- IIOPS

com.sybase.jaguar.listener.security

Description: For listeners that use a tunnelled SSL protocol (such as HTTPS or IIOPS), specifies the security profile used to configure the SSL session settings.

Syntax: The name of the security profile, which must match the com.sybase.jaguar.security.name property of the security profile.

See also: Security properties on page 500

com.sybase.jaguar.listener.solaris.tli.maxoutcon

Description: When running on Solaris, specifies the pool size for outstanding connection requests.

Syntax: Values must be a positive integer less than or equal to 4096. If this property is not set, the default is 128. Values greater than 4096 are truncated to 4096 to avoid excessive memory allocation at startup.

Usage

This property is used on Solaris only. When the server is very busy, all available threads may be in use when a connect request arrives. These pending connect requests are pooled until they can be handled. If the pool size is too small, client connection requests may time out before the server can handle the request.

You can configure different request pool sizes for different protocols. For example, if the server is handling mostly HTTP requests, you can increase the request pool size for the HTTP listener while leaving the IIOP request pool size at a low value.

The connection request pool size affects the server memory requirements:

```
mem = entries * 20K
```

That is, each entry requires about 20K of memory reserved at server startup.

com.sybase.jaguar.listener.type

Description

For TDS listeners, specifies whether this listener accepts Open Server client connections.

Syntax

The only allowable value for this property is `os`. If the property is set with no value, or not set, the listener will not accept Open Server client connections.

Usage

EAServer allows you to install Open Server event handlers so that you can migrate legacy Sybase Open Server applications to EAServer. The event handlers are called only for connections to a listener that uses the TDS protocol and that has the com.sybase.jaguar.listener.type property set to "os". For more information on this feature, see Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*.

See also

com.sybase.jaguar.listener.protocol

Log profile properties

Description

Log profile properties are stored as *.props* files in the EAServer *Repository/LogProfile* directory. This release provides limited facilities for editing profiles in EAServer Manager.

You can install a log profile in a server by setting the server properties com.sybase.jaguar.server.logging.profile.debug and com.sybase.jaguar.server.logging.profile.prod.

com.sybase.jaguar.logprofile.description

Description — Specifies an optional description of the log profile.

Syntax — The text of the description.

com.sybase.jaguar.logprofile.name

Description — Specifies the log profile name.

Syntax — The name, which must match the base name of the properties file. For example, if the file is *myprod.props*, the name must be `myprod`.

com.sybase.jaguar.logprofile.subsystem

Description — Specifies the logging subsystem to be used.

Syntax — One of the following:

Value	To indicate
`eas`	The built-in EAServer logging system, as used in EAServer versions earlier than 5.0. To configure the output, set the additional properties described in Log profile Java Logging subsystem properties.
`l4j`	Apache Log4j. To configure the output, set the additional properties described in Log profile Log4j subsystem properties.
`jdk`	The java.util.logging system, available in JDK 1.4 and later JDK versions. This subsystem cannot be used in servers running JDK 1.3 or earlier. To use this subsystem, configure the additional properties described in Log profile Java Logging subsystem properties.

Log profile EAS subsystem properties

Description — These properties configure the output from a logging profile that uses the built-in EAS logging subsystem. The properties can be logically grouped as category properties, handler properties, and formatter properties.

Categories A *category* is a logical name used to categorize log messages. Internally in EAServer, different category names are used by different subsystems such as the servlet engine, as listed in "Category names" on page 60.

Categories can be arranged hierarchically by setting the parent property for handlers that should inherit settings from another handler. There is also a root category that configures the root-level settings. If you configure a category to inherit properties, but do not specify a parent, the root category settings are inherited.

Table B-8 lists the properties that configure a category. In the log profile, you define a category by configuring properties that begin with category.<*cat-name*>, where *cat-name* is replaced by the category name. For example, category.com.sybase.level configures the log level for the `com.sybase` category.

Table B-8: Category properties

Property name	Specifies
category.<root>.level	The log level for the root category.
category.<root>.handler	The handler for the root category.
category.<cat-name>.description	Specifies an optional description of the category.
category.<cat-name>.level	The log level for the category named <*cat-name*>.
category.<cat-name>.handler	The handler for the category named <*cat-name*>.
category.<cat-name>.parent	The parent handler.
category.<cat-name>.useparenthandlers	Whether to inherit settings from the parent handler.
category.<cat-name>.resourcebundlename	The resource bundle name for the category <*cat-name*>. Specify the name of a Java resource bundle name containing localized messages that are logged from Java code. If this property is not set, the default resource bundle is the class ResourceBundle in the package with the same name as the category. For example, the default resource bundle for the category com.sybase.jaguar.server is com.sybase.jaguar.server.ResourceBundle.

Handlers Handlers define how messages are logged. For example, you set properties to specify whether the output goes to the console or to a file, the log file name, whether to truncate log files on start-up, and so forth. To define a handler, specify a logical name when setting the handler properties in Table B-9 with <*handler-name*> replaced in the property name by the handler name.

Table B-9: Handler properties

Property name	Specifies
handler.<handler-name>.description	An optional description
handler.<handler-name>.type	The log output type

Property name	Specifies
handler.<handler-name>.consoletype	For console output, whether to use standard output or standard error
handler.<handler-name>.filename	For file output, the log file name
handler.<handler-name>.truncate	For file output, whether to truncate the log file at start-up
handler.<handler-name>.maxsize	For file output, the maximum size
handler.<handler-name>.rotate	For file output, whether to rotate log files at start-up and when the maximum size is reached
handler.<handler-name>.formatter	The formatter name to use
handler.<handler-name>.archive	Whether to archive files at start-up and when the maximum size is reached
handler.<handler-name>.archive.filename	The location and naming pattern for archived log files
handler.<handler-name>.archive.compress	When archiving, whether to compress the log file
handler.<handler-name>.serverhost	When writing to a TCP socket, the host name for the socket connection
handler.<handler-name>.serverport	When writing to a TCP socket, the port number for the socket connection

Formatters Formatters specify the formatting pattern for logged messages. To define a formatter, specify a logical name when setting the handler properties in Table B-10, with <*formatter-name*> replaced in the property name by the formatter name.

Table B-10: Formatter properties

Property name	Specifies
formatter.<formatter-name>.description	An optional description
formatter.<formatter-name>.dateformat	The format for timestamps in log messages
formatter.<formatter-name>.messageformat	The format pattern for the message text

See also Log profile properties, com.sybase.jaguar.logprofile.subsystem

category.<cat-name>.description

Description Specifies an optional description of the category named <*cat-name*>.

Syntax The text of the description.

See also “Categories” on page 462.

category.<cat-name>.handler

Description Specifies the handler for the category named <*cat-name*>.

Syntax The handler name. “Handlers” on page 462 describes how to define a handler.

See also “Categories” on page 462.

category.<cat-name>.level

Description Specifies the log level for the category named <*cat-name*>.

Syntax One of the error levels listed in Table B-11. The error level specifies which messages are logged. Only messages of the specified security level or greater are logged. Table B-11 lists the levels in ascending order of severity.

Table B-11: EAS Log subsystem error levels

Level	To indicate
ALL	All messages are logged.
FINE	Debug messages are logged.
FINER	Same effect as FINE.
FINEST	Same effect as FINE.
CONFIG	A configuration error has been detected. You should correct the problem.
INFO	An informational or status message. For example, the name server has finished binding components.
WARN	Warning messages are logged. For example, if the server is in a cluster and other members are not found.
ERROR	An error has occurred that prevents completion of a requested action. For example, a component has thrown an uncaught exception and its transaction is being rolled back.
FATAL	An error has occurred that indicates the server should terminate.
OFF	No messages are logged.

See also "Categories" on page 462, category.<root>.level

category.<cat-name>.parent

Description Specifies the parent of the category named <*cat-name*>.

Syntax The name of the parent category. If not specified, the default is the root category.

See also "Categories" on page 462, category.<cat-name>.useparenthandlers

category.<cat-name>.resourcebundlename

Description Specifies the name of the Java resource bundle containing localized messages.

Syntax The full Java class name of the resource bundle, which must extend java.util.ResourceBundle. Do not include language or country code extensions.

See also "Categories" on page 462

category.<cat-name>.useparenthandlers

Description: For the category named <*cat-name*>, specifies whether to inherit settings from the parent handler. The parent is specified by the category.<cat-name>.parent property. If this property is not set, the parent is the root category.

Syntax: `true` or `false`.

See also: "Categories" on page 462, category.<cat-name>.parent

category.<root>.level

Description: Specifies the log level for the root category.

Syntax: Same as category.<cat-name>.level.

See also: category.<root>.handler, category.<cat-name>.level, "Categories" on page 462

category.<root>.handler

Description: Specifies the handler for the root message category.

Syntax: The handler name. "Handlers" on page 462 describes how to define a handler.

See also: category.<root>.level, category.<cat-name>.handler, "Categories" on page 462

formatter.<formatter-name>.dateformat

Description: Specifies the timestamp format for the formatter named <*formatter-name*>.

Syntax: The pattern for the timestamps embedded in log messages, as converted to strings by java.text.SimpleDateFormat. For details, see the the DateFormat API documentation at http://java.sun.com/j2se/1.4.1/docs/api/java/text/SimpleDateFormat.html.

See also: "Formatters" on page 463, formatter.<formatter-name>.messageformat

formatter.<formatter-name>.description

Description — Specifies an optional description for the formatter named <*formatter-name*>.

Syntax — The text of the description.

See also — “Formatters” on page 463

formatter.<formatter-name>.messageformat

Description — Specifies the message format for the formatter named <*formatter-name*>.

Syntax — The pattern for the message text. You can use the placeholders in Table B-12 to indicate the position of the message parts.

Table B-12: Message format placeholders

Placeholder	Represents
`%LN`	The logging category name
`%MC`	Message code (number)
`%ML`	Message level (severity)
`%MT`	Message text
`%SN`	Sequence number
`%SF`	Source file name
`%SL`	Line number in the source file
`%SM`	Method name in the source file
`%TI`	Thread ID
`%TH`	Exception thrown (if available)
`%TS`	The timestamp, formatted as specified by the formatter.<formatter-name>.dateformat property
`%NL`	A line break

See also — “Formatters” on page 463,
formatter.<formatter-name>.dateformat

handler.<handler-name>.archive

Description — For the handler named <*handler-name*>, when writing to a file, specifies whether the previous log file should be archived when restarting servers or when the maximum size is reached.

Syntax — `true` or `false`. The default is `false`. If you enable archiving, specify the archive file name by setting handler.<handler-name>.archive.filename.

See also — "Handlers" on page 462,
handler.<handler-name>.type,
handler.<handler-name>.maxsize,
handler.<handler-name>.archive.filename,
handler.<handler-name>.archive.compress

handler.<handler-name>.archive.compress

Description — For the handler named <*handler-name*>, when file archiving is enabled, specifies whether to compress archived files.

Syntax — `true` or `false`. The default is `false`. If you enable compression, the resulting archive name is the value of handler.<handler-name>.archive.filename plus the ".zip" extension.

See also — "Handlers" on page 462,
handler.<handler-name>.archive
handler.<handler-name>.archive.filename

handler.<handler-name>.archive.filename

Description — For the handler named <*handler-name*>, when file archiving is enabled, specifies the archive file name.

Syntax — The full path to the archive file, using '/' (forward slash) as the directory separator. You can use the placeholders described in Table B-13 on page 469.

See also — "Handlers" on page 462,
handler.<handler-name>.archive,
handler.<handler-name>.archive.compress

handler.<handler-name>.consoletype

Description — For the handler named <*handler-name*>, when writing to the console, specifies whether to write to standard error or standard output.

Syntax — One of the following:

Value	To indicate
`stderr`	Standard error
`stdout`	Standard output

See also "Handlers" on page 462, handler.<handler-name>.type

handler.<handler-name>.description

Description Specifies the output type for the handler named <*handler-name*>.

Syntax The descriptive text.

See also "Handlers" on page 462

handler.<handler-name>.filename

Description For the handler named <*handler-name*>, when writing to a file, specifies the file name.

Syntax The full path to the output file, using '/' (forward slash) as the directory separator. You can use the placeholders described in Table B-13:

Table B-13: File name and path placeholders

Placeholder	Specifies
`${JAGUAR}`	The EAServer installation directory (specified by the JAGUAR environment variable)
`${JAGEXEDIR}`	The location of binaries in the EAServer installation (*$JAGUAR/bin* on most platforms)
`${JAGSRVNAME}`	The server name, as displayed in EAServer Manager
`%t`	The time of day when the file was created, formatted as HHMMSS
`%d`	The date (month, day of month, and year) when the file was created, formatted as DDMMYYYY

See also "Handlers" on page 462,
handler.<handler-name>.type

handler.<handler-name>.formatter

Description For the handler named <*handler-name*>, specifies the formatter name.

Syntax The formatter name. See "Formatters" on page 463.

See also "Handlers" on page 462,
"Formatters" on page 463

handler.<handler-name>.maxsize

Description — For the handler named <*handler-name*>, when writing to a file, specifies the maximum size the file can reach before rotation or archiving occurs.

Syntax — To specify the size in kilobytes:

*n*k

To specify the size in megabytes:

*n*m

Where *n* is a positive integer. To specify an unlimited size, use `-1` (the default).

Usage — If you specify a maximum size, you should enable rotation or archiving by setting the handler.<handler-name>.rotate or handler.<handler-name>.archive properties, respectively.

See also — "Handlers" on page 462,
handler.<handler-name>.type,
handler.<handler-name>.archive,
handler.<handler-name>.rotate

handler.<handler-name>.rotate

Description — For the handler named <*handler-name*>, when writing to a file, specifies whether the previous log file should be renamed when restarting servers or when the maximum size is reached.

Syntax — `true` or `false`. The default is `false`. If you enable rotation, the log file is renamed with a sequential numeric extension when a new file is started. For example, if the file is *Jaguar.log*, previous versions are named *Jaguar.log.1*, *Jaguar.log.2*, and so forth.

See also — "Handlers" on page 462,
handler.<handler-name>.type,
handler.<handler-name>.archive.filename,
handler.<handler-name>.maxsize

handler.<handler-name>.serverhost

Description — For the handler named <*handler-name*>, when writing to a TCP socket, specifies the host name for the socket connection.

Syntax — The server host name or IP address.

See also
"Handlers" on page 462, ,
handler.<handler-name>.serverport
handler.<handler-name>.type

handler.<handler-name>.serverport

Description
For the handler named <*handler-name*>, when writing to a TCP socket, specifies the port number for the socket connection.

Syntax
The server port number.

See also
"Handlers" on page 462, ,
handler.<handler-name>.serverhost
handler.<handler-name>.type

handler.<handler-name>.truncate

Description
For the handler named <*handler-name*>, when writing to a file, specifies whether the previous log file should be truncated when restarting servers.

Syntax
`true` or `false`. The default is `false`.

See also
"Handlers" on page 462,
handler.<handler-name>.type

handler.<handler-name>.type

Description
Specifies an optional description for the handler named <*handler-name*>.

Syntax
One of the following:

Value	To indicate
`file`	Output to a text file
`console`	Output to the console (standard error or standard out)
`tcp`	Output to a TCP socket

See also
"Handlers" on page 462

Log profile Java Logging subsystem properties

Description — These properties configure the output from a logging profile that uses the JDK java.util.logging package as its logging system. All properties in the logging profile are applied to the Log4j configuration. Only those properties used in the preconfigured JDK log profiles are documented here. For information on other properties, see the Java Logging package documentation at http://java.sun.com/j2se/1.4.1/docs/guide/util/logging/overview.html

See also — Log profile properties, com.sybase.jaguar.logprofile.subsystem

.level

Description — Specifies the log level for the root logger.

Syntax — Same as for <logger>.level.

<logger>.level

Description — Defines the logging level for logger named `<logger>`. The logger names used internally in EAServer are the same as for Log4j, specified in Table B-8 on page 462.

Syntax — One of the levels listed in Table B-14 on page 473, as defined in java.util.logging.Level. Levels are listed in order of ascending severity. Only messages with severity greater than the specified level are logged.

Table B-14: java.util.logging error levels

Level	To indicate
ALL	All messages are logged.
FINE	Debug messages are logged.
FINER	Same effect as FINE.
FINEST	Same effect as FINE.
CONFIG	A configuration error has been detected. You should correct the problem.
INFO	Warning and informational messages are logged. Informational messages include status information, such as the name server has finished binding components. An example warning would be that the server is in a cluster and other members are not found.
WARNING	An error has occurred that prevents completion of a requested action. For example, a component has thrown an uncaught exception and its transaction is being rolled back.
SEVERE	An error has occurred that indicates the server should terminate.
OFF	No messages are logged.

Usage

Loggers allow you to associate logical names with categories of messages. For example, `com.sybase.jaguar.servlet` is the logger name for messages from the EAServer servlet engine. Table B-8 on page 462 lists the logger names used internally by EAServer. Loggers use a name-based hierarchy, similar to the Java package naming convention, and inherit settings from their parent. For example, a logger named `com.sybase.jaguar.servlet` inherits the settings for `com.sybase`, unless you explicitly configure `com.sybase.jaguar`. If a logger is not configured explicitly, it inherits the root logger setting configured by the .level property.

The same names are used for Log4j logger names, EAS log subsystem categories, and java.util.logging logger names.

See also

.level,
java.util.logging.ConsoleHandler.level,
java.util.logging.FileHandler.level

handlers

Description

Specifies handlers for the log output.

Syntax — A comma-separated list of classes that implement java.util.logging.Handler. For example:

```
handlers=java.util.logging.FileHandler,
java.util.logging.ConsoleHandler
```

See also — java.util.logging.ConsoleHandler.formatter, java.util.logging.ConsoleHandler.level, java.util.logging.FileHandler.formatter, java.util.logging.FileHandler.level, java.util.logging.FileHandler.pattern

java.util.logging.ConsoleHandler.formatter

Description — If using handler java.util.logging.ConsoleHandler, the formatter class to be used.

Syntax — The handler class name. The class must implement java.util.logging.Formatter.

java.util.logging.ConsoleHandler.level

Description — If using handler java.util.logging.ConsoleHandler, the log level for file output to be logged.

Syntax — Same as <logger>.level. The most restrictive of the <logger>.level and the console handler setting is used.

See also — <logger>.level

java.util.logging.FileHandler.formatter

Description — If using handler java.util.logging.FileHandler, the formatter class to be used.

Syntax — The handler class name. The class must implement java.util.logging.Formatter.

java.util.logging.FileHandler.level

Description — If using handler java.util.logging.FileHandler, the log level for file output to be logged.

Syntax	Same as <logger>.level. The most restrictive of the <logger>.level and the file handler setting is used. For example, if the logger level is INFO, and the file handler setting is WARNING, only WARNING messages are logged by the file handler. If the logger level is INFO, and the file handler setting is ALL, the effective level is WARNING.
See also	<logger>.level

java.util.logging.FileHandler.pattern

Description	The pattern (template) for the output file name.
Syntax	See the FileHandler API documentation at http://java.sun.com/j2se/1.4.1/docs/api/java/util/logging/FileHandler.html.

Log profile Log4j subsystem properties

Description	These properties configure the output from a logging profile that uses the Apache Log4j logging system. All properties in the logging profile are applied to the Log4j configuration. Only those properties used in the preconfigured Log4j profiles are documented here. For information on other properties, see the Apache Log4j Documentation at http://jakarta.apache.org/log4j/docs/api/overview-summary.html.
See also	Log profile properties, com.sybase.jaguar.logprofile.subsystem

log4j.rootLogger

Description	Configures the Log4j root log level and appenders.
Syntax	*level*[, *appenders*]
	Where *level* is one of the error levels listed in Table B-15, as defined in class org.apache.log4j.Level. The error level specifies which messages are logged. Only messages of the specified level or greater severity are logged. Table B-15 lists the levels in ascending order of severity.

Table B-15: Log4j error levels

Level	To indicate
ALL	All messages are logged.
DEBUG	Debug messages are logged.
INFO	Informational messages are logged, for example, a listener has been established or the name service has finished binding installed components.
WARN	Warning messages are logged. For example, the server is in a cluster and other members are not found.
ERROR	An error has occurred that prevents completion of a requested action. For example, a component has thrown an uncaught exception and its transaction is being rolled back.
FATAL	An error has occurred that indicates the server should terminate.
OFF	No messages are logged.

appenders specifies an optional comma-separated list of appenders. In Log4j, an *appender* specifies how messages are logged, for example, to the console or to a file. Each appender must be configured by setting additional properties, prefixed with log4j.appender.*name*, where *name* is the appender name.

See also log4j.logger.<logger-name>, log4j.logger.<logger-name>

log4j.logger.<logger-name>

Description Configures the log output for the logger specified by <logger-name>.

Syntax Same as for log4j.rootLogger.

Usage This property configures the log output properties for a logger name used in code. Logger names are a logical means to categorize messages, and setting this property allows you to configure the output for messages written to the specified logger. EAServer uses the logger names listed in Table B-8 on page 462. The same names are used for Log4j logger names, EAS log subsystem categories, and java.util.logging logger names.

You can add additional properties to configure logger names used in your own code. If not configured explicitly, loggers inherit configurations from their parent in the prefix-based name hierarchy. For example, a logger named `com.sybase.jaguar` inherits the settings for `com.sybase`, unless you explicitly configure `com.sybase.jaguar`. If there is no logger configured with a prefix name, loggers inherit the root logger settings specified by log4j.rootLogger.

For example, this setting configures messages written to the logger named `com.sybase.jaguar.servlet`. As a result, only messages with severity level INFO or greater are logged, and messages to this logger go to the output configured for the appender `eas_servlet`:

```
log4j.logger.com.sybase.jaguar.servlet=INFO, eas_servlet
```

See also log4j.rootLogger, log4j.additivity.<logger-name>

log4j.additivity.<logger-name>

Description Configures *additivity* for the logger, that is, whether messages written to the logger go only to destinations configured explicitly for this logger or also go to destinations configured for the parent in the hierarchy.

Syntax `true` or `false`. If not specified, the default is `true`.

Examples These settings configure the output for the `com.sybase` logger and the `com.sybase.jaguar.servlet` logger to use separate destinations (specified in the configuration of the `eas` and `eas_servlet` appenders, respectively). The additivity is false for the `com.sybase.jaguar.servlet` logger. If the log4j.additivity.com.sybase.jaguar.servlet property were not set to false, the servlet log output would go to both destinations:

```
log4j.rootLogger=INFO, eas
log4j.logger.com.sybase=INFO
log4j.logger.com.sybase.jaguar.servlet=INFO, eas_servlet
log4j.additivity.com.sybase.jaguar.servlet=false
```

See also log4j.logger.<logger-name>

log4j.appender.<name>

Description Specifies the class that writes messages for appender <*name*>, where <*name*> is an appender name specified in a log4j.rootLogger or log4j.logger.<logger-name> setting.

Syntax The full name of an instantiable Java class that implements the org.apache.log4j.Appender interface. Examples used in the default EAServer Log4j configuration include:

Appender class	Description
`org.apache.log4j.FileAppender`	Logs messages to a text file

Appender class	Description
`org.apache.log4j.ConsoleAppender`	Logs messages to the console (standard output or standard input)

Log4j includes other appenders and you can implement your own. See the Apache Log4j Documentation at http://jakarta.apache.org/log4j/docs/api/overview-summary.html for more information. If you implement your own classes, make sure they are in the CLASSPATH and BOOTCLASSPATH for the EAServer process.

See also log4j.appender.<appender-name>.Append, log4j.appender.<appender-name>.File, log4j.appender.<appender-name>.layout, log4j.appender.<appender-name>.layout.ConversionPattern, log4j.appender.<appender-name>.target

log4j.appender.<appender-name>.Append

Description For appenders that use `org.apache.log4j.FileAppender`, specifies whether the output file is opened for appending or truncated.

Syntax `true` or `false`. A value of `true` specifies that the output file must be appended to. If you specify false, the server log files are truncated each time you restart the server.

See also log4j.appender.<name>

log4j.appender.<appender-name>.File

Description For appenders that use `org.apache.log4j.FileAppender`, specifies the output file name and location.

Syntax The full path to the output file, using '/' (forward slash) as the directory separator. You can use the placeholders described in Table B-16:

Table B-16: File name and path placeholders

Placeholder	Specifies
`${JAGUAR}`	The EAServer installation directory (specified by the JAGUAR environment variable)
`${JAGEXEDIR}`	The location of binaries in the EAServer installation (*$JAGUAR/bin* on most platforms)
`${JAGSRVNAME}`	The server name, as displayed in EAServer Manager

See also log4j.appender.<name>

log4j.appender.<appender-name>.layout

Description Specifies the layout class for the appender.

Syntax The full name of an instantiable Java class that implements org.apache.log4.Layout, for example:

org.apache.log4j.PatternLayout

See also log4j.appender.<appender-name>.layout.ConversionPattern

log4j.appender.<appender-name>.layout.ConversionPattern

Description For appenders that use layout org.apache.log4j.PatternLayout, specifies the pattern to format the log message contents.

Syntax See the description of the pattern string in the org.apache.log4j.PatternLayout documentation at http://jakarta.apache.org/log4j/docs/api/org/apache/log4j/PatternLayout.html.

See also log4j.appender.<appender-name>.layout

log4j.appender.<appender-name>.target

Description For appenders that use `org.apache.log4j.ConsoleAppender`, specifies whether to write to standard output or standard error.

Syntax One of the following values:

Value	Specifies
`System.out`	Standard output.
`System.err`	Standard error.

See also log4j.appender.<name>

Method properties

Description Method property names begin with `com.sybase.jaguar.method`. In EAServer Manager, configure method properties in the Method Properties dialog box, and in the Interfaces folder beneath a component that uses the method.

Method properties are configured on a per-component basis. An interface used by two components may have different method properties for each component.

See also com.sybase.jaguar.component.methods

com.sybase.jaguar.method.flags

Description Specifies flags to enable nondefault method behavior.

Syntax A comma-separated list of the following:

Flag	To indicate
ks	This flag must be present if the method returns a key sequence, that is, the method is an entity finder method that returns multiple keys.
ro	The method is read-only. For stateful or entity components only. Specifies whether the method can change the component's state. This setting allows the server to omit processing that is otherwise required to save a component's state. For example, the ejbStore method is not called for an EJB entity bean if a read-only method has been invoked.

Usage In EAServer Manager, you can set the "ro" flag with the Read only check box on the General tab in the Method Properties dialog box. To set other flags, use the Advanced tab.

com.sybase.jaguar.method.iso_level

Description For EJB 1.0 components, specifies the isolation level for transactions begun by the component's methods.

Syntax See com.sybase.jaguar.component.iso_level.

com.sybase.jaguar.method.monitor

Description	Assigns this method to a thread monitor.
Syntax	Specify the monitor name.
Usage	If you change this setting, you must regenerate and recompile the component skeleton for the change to take effect.
See also	Thread monitor properties, com.sybase.jaguar.component.monitor

com.sybase.jaguar.method.name

Description	Specifies the method name.
Syntax	*package*/*component*/*method* Where: • *package* is the package name. • *component* is the component name. • *method* is the IDL method name. For EJB home interface methods, specify the method name with prefix "ejb" and the first letter capitalized. For example, specify ejbCreate for the home interface create method.

com.sybase.jaguar.method.roles

Description	For non-EJB components, specifies role memberships required to execute the method.
Syntax	*role1*, *role2*, ... Where *role1*, *role2*, and so forth are role names defined in the repository.
Usage	In EAServer Manager, set this property using the File menu for the Roles folder displayed below the method icon. Roles are attached to EAServer packages, components, and methods. Attaching a role to a package controls access to all non-EJB components in the package. Attaching a role to a component constrains access to all methods in the component's interfaces. Attaching a role to a method constrains access to that method. For EJB components, configure access control at the method level by setting the method property com.sybase.jaguar.method.security-roles for each method.

See also com.sybase.jaguar.component.roles, com.sybase.jaguar.package.roles, com.sybase.jaguar.method.security-roles

com.sybase.jaguar.method.runasidentity

Description For EJB 1.0 components, specifies an identity name used for intercomponent calls if the com.sybase.jaguar.method.runasmode property is "specified."

Syntax Same as for the com.sybase.jaguar.component.runasidentity component property.

Usage In EAServer Manager, set this property using the Run As Mode tab in the Method Properties dialog box.

See also com.sybase.jaguar.method.runasmode, com.sybase.jaguar.component.runasidentity

com.sybase.jaguar.method.runasmode

Description For EJB 1.0 components, specifies the user identity that is assumed for intercomponent calls.

Syntax Same as for the com.sybase.jaguar.component.runasmode component property.

Usage In EAServer Manager, set this property using the Run As Mode tab in the Method Properties dialog box.

See also com.sybase.jaguar.method.runasidentity, com.sybase.jaguar.component.runasmode

com.sybase.jaguar.method.security-roles

Description For EJB components, specifies role memberships required to execute the method.

Syntax A comma-separated list of role reference names. Each name must exist in the repository to be mapped to an EAServer role name by the package or application properties.

If this property is not set, the behavior depends on the server property com.sybase.jaguar.server.ejb.role.default. If the property is set, the user must be a member of one of the required EAServer roles to invoke the method.

Usage: In EAServer Manager, set this property on the Permissions tab in the Method Properties dialog box.

See also: com.sybase.jaguar.application.security-roles, com.sybase.jaguar.application.security-role.<j2ee-role>, com.sybase.jaguar.package.security-roles, com.sybase.jaguar.package.security-role.<j2ee-role>

com.sybase.jaguar.method.tx_type

Description: Specifies how this method participates in transactions.

Syntax: Same as for the com.sybase.jaguar.component.tx_type component property. If the method property is not set, the component setting is used.

Usage: In EAServer Manager, set this property on the Transactions tab in the Method Properties dialog box.

See also: com.sybase.jaguar.component.tx_type

Package properties

Description: Package property names begin with `com.sybase.jaguar.package`. A package allows you to group related components together for the purpose of access control or configuration. In EAServer Manager, configure package properties in the Package Properties dialog box.

com.sybase.jaguar.description

Description: Specifies a text description of the package.

Syntax: *desc*

Where *desc* is the descriptive text.

Usage: In EAServer Manager, set this property using the Description field on the General tab of the Package Properties dialog box.

com.sybase.jaguar.package.application

Description	Specifies the application that this package is installed in, if any.
Syntax	The name of the application. If not set, the package is not in an application and may be installed directly in one or more servers.
Usage	A package may be installed in one application or one or more servers. A package installed in an application may not be installed directly in a server or in another application. In EAServer Manager, set this property by installing the package in an application or removing the package from an application.
See also	com.sybase.jaguar.application.packages

com.sybase.jaguar.package.classloaderpolicy

Description	Specifies how the custom class loader (version 2) resolves version conflicts when you specify the same class at multiple levels in the class loader hierarchy.
Syntax	Same as com.sybase.jaguar.application.classloaderpolicy.
See also	com.sybase.jaguar.server.jvm.classloader, com.sybase.jaguar.application.classloaderpolicy, com.sybase.jaguar.component.classloaderpolicy, com.sybase.jaguar.webapplication.classloaderpolicy Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.package.code.set

Description	Specifies the default for the com.sybase.jaguar.component.code.set component property.
Syntax	Same as com.sybase.jaguar.component.code.set. The default is the server property com.sybase.jaguar.server.code.set.
Usage	In EAServer Manager, set this property using the Advanced tab in the Package Properties dialog box.
See also	com.sybase.jaguar.component.code.set, com.sybase.jaguar.server.code.set

com.sybase.jaguar.package.DOMfactory

Description	Specifies the class name for a custom DOM XML parser factory class.
Syntax	The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.package.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.
See also	com.sybase.jaguar.package.SAXfactory, com.sybase.jaguar.package.XSLTfactory

com.sybase.jaguar.package.files

Description	Specifies files to be included when the package is exported to a Jaguar JAR archive or replicated to another installation using synchronization.
Syntax	Same as for com.sybase.jaguar.applicationclient.files. A package archive includes files for installed components, plus those specified by this property.
See also	com.sybase.jaguar.package.files.corbastubs, com.sybase.jaguar.package.files.ejbstubs, com.sybase.jaguar.component.files

com.sybase.jaguar.package.files.corbastubs

Description	Specifies the files that implement Java/CORBA stubs for the components in the package.
Syntax	Set when Java/CORBA stubs are generated. Do not modify.
See also	com.sybase.jaguar.package.files, com.sybase.jaguar.package.files.ejbstubs, com.sybase.jaguar.component.files.corbastubs

com.sybase.jaguar.package.files.ejbstubs

Description	Specifies the files that implement EJB stubs for the components in the package.
Syntax	Set when EJB stubs are generated. Do not modify.
See also	com.sybase.jaguar.package.files, com.sybase.jaguar.package.files.corbastubs, com.sybase.jaguar.component.files.ejbstubs

com.sybase.jaguar.package.java.classes

Description	Specifies Java classes and JAR files to be loaded by the package's custom class loader.
Syntax	Same as for com.sybase.jaguar.application.java.classes.
Usage	See "Custom class lists for packages, applications, or servers" in Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*. In EAServer Manager, set this property using the Java Classes tab in the Package Properties dialog box.
See also	com.sybase.jaguar.component.java.classes, com.sybase.jaguar.application.java.classes, com.sybase.jaguar.server.java.classes

com.sybase.jaguar.package.name

Description	Specifies the package name.
Syntax	*pack-name* Where *package-name* is the application name.
Usage	In EAServer Manager, the package name is set when creating or importing a package and cannot be changed.

com.sybase.jaguar.package.roles

Description	Specifies roles that a user must belong to in order to invoke the components (other than EJB components) installed in this package.
Syntax	*role1*, *role2*, ... Where *role1*, *role2*, and so forth are role names defined in the repository.
Usage	In EAServer Manager, set this property using the File menu for the Roles folder displayed below the package icon. Roles are attached to EAServer packages, components, and methods. Attaching a role to a package controls access to all non-EJB components in the package. Attaching a role to a component constrains access to all methods in the component's interfaces. Attaching a role to a method constrains access to that method.

For EJB components, configure access control at the method level by setting the method property com.sybase.jaguar.method.security-roles for each method.

See also com.sybase.jaguar.component.roles, com.sybase.jaguar.method.roles, com.sybase.jaguar.method.security-roles

com.sybase.jaguar.package.runasidentity.<id>

Description Maps an identity name used in EJB (1.0 and 2.0) component configuration to an identity defined in the EAServer repository.

Syntax Specify the identity referenced in EJB component configuration in the property name, for example:

```
com.sybase.jaguar.package.runasidentity.ejbFooId
```

Specify the mapped EAServer identity as the value, for example:

```
fooId
```

Usage In EAServer Manager, you can set and view this property using the Run As Identity tab in the Component Properties dialog box, when the component type is an EJB 1.0 or 2.0 component that is installed in this package.

See also Security properties on page 500, com.sybase.jaguar.component.runasidentity, com.sybase.jaguar.component.security.runasidentity

com.sybase.jaguar.package.SAXfactory

Description Specifies the class name for a custom SAX XML parser factory class.

Syntax The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.package.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.

See also com.sybase.jaguar.package.DOMfactory, com.sybase.jaguar.package.XSLTfactory

com.sybase.jaguar.package.security-role.<j2ee-role>

Description — Specifies a mapping from a J2EE role name used in the package to a role defined in the EAServer repository.

Syntax — com.sybase.jaguar.package.security-role.*j2ee-role=jag-role*

Where:

- *j2ee-role* is the role name used in the application and listed in the com.sybase.jaguar.package.security-roles property.
- *jag-role* is the role name defined in the EAServer repository.

Usage — In EAServer Manager, set this property using the Role Mapping tab in the Package Properties dialog box.

Role names may also be specified at the component or application level.

See also — com.sybase.jaguar.package.security-roles, com.sybase.jaguar.application.security-roles, com.sybase.jaguar.component.security-role-ref.<j2ee-role-ref>, com.sybase.jaguar.component.security-role-refs

com.sybase.jaguar.package.security-roles

Description — Specifies logical J2EE role names used in the package.

Syntax — *role1*, *role2*, ...

Where *role1*, *role2*, and so forth are of the form:

(description=*role-desc*, name=*role-name*)

Where *role-desc* is an optional description of the role, and *role-name* is the name used in the application.

Usage — In EAServer Manager, set this property using the Role Mapping tab in the Package Properties dialog box. Each J2EE role name must be mapped to an EAServer role name by setting or creating the corresponding com.sybase.jaguar.package.security-role.<j2ee-role> property.

J2EE role names may also be specified at the component or application level.

See also — com.sybase.jaguar.package.security-role.<j2ee-role>, com.sybase.jaguar.application.security-role.<j2ee-role>, com.sybase.jaguar.component.security-role-ref.<j2ee-role-ref>, com.sybase.jaguar.component.security-role-refs

com.sybase.jaguar.package.XSLTfactory

Description — Specifies the class name for a custom XSLT XML parser factory class.

Syntax — The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.package.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.

See also — com.sybase.jaguar.component.DOMfactory, com.sybase.jaguar.component.SAXfactory

schema:<schema-name>

Description — Specifies the EJB 2.0 (or later) component that represents a schema name used in EJB-QL queries.

Syntax — schema:*schema-name=package/component*

Where:

Variable	Specifies
schema-name	The schema name used in EJB-QL queries
package	The package that contains the component
component	The component name

If not set, the default is ***this-package*/*schema-name***, where *this-package* is the package name, and *schema-name* is the schema name used in the EJB-QL query.

Usage — EJB-QL queries are used in EJB 2.0 entity beans that require container-managed persistence.

See also — Chapter 27, "Creating Entity Components," in the *EAServer Programmer's Guide*

Resource environment reference properties

Description — For application clients, EJB components, and Web applications, you can define resource environment references and their properties. Resource environment references specify aliased JNDI names for JMS message queues and topics used by the application code.

Resource reference syntax

Resource environment reference properties are of the form:

res-env-ref1, *res-env-ref2*, ...

Where *res-env-ref1*, *res-env-ref2*, and so forth are of the form:

(description=*desc*,jag-resource=*link*,
res-env-ref-type=*type*,res-env-ref-name=*jndi-name*)

Where:

- *desc* is an optional comment describing how the entry is to be set.
- *link* is the name of either a queue connection factory or a topic connection factory, depending on the value of res-env-ref-type:
  - For javax.jms.Queue – queue connection factory
  - For javax.jms.Topic – topic connection factory
- *type* is the resource type, which can be either:
  - javax.jms.Queue
  - javax.jms.Topic
- *jndi-name* is the partial JNDI name, relative to *java:comp/env*. Use the prefix *jms/* for JMS reference. For example, if your code refers to *java:comp/env/jms/MyQueue*, enter `jms/MyQueue`.

See also

com.sybase.jaguar.applicationclient.resource-env-ref,
com.sybase.jaguar.component.resource-env-ref,
com.sybase.jaguar.webapplication.resource-env-ref

Resource manager properties

Description

Resource manager properties represent global properties for all native connection libraries. The properties are stored in the database property files, located in EAServer's */Repository/ResourceManager* directory. The property names begin with com.sybase.jaguar.resourcemanager. Table B-17 lists the database property file names for each connection library type.

Table B-17: Database property files

Connection library type	Database property file
Sybase Client Library 11.0	CTLIB_110.props
Oracle OCI 7.x	OCI_7.props
Oracle OCI 8.x	OCI_8.props

Connection library type	Database property file
Oracle OCI 9.x	OCI_9.props

Each property file is preconfigured with the connection library and XA-Library names for these operating systems: Solaris, Windows, HP-UX, AIX, and Linux. Table B-18 lists the values for Solaris and Windows.

Table B-18: XA resource libraries and property files

Connection library type	XA-Library for Solaris	XA-Library for Windows	Connection library for Solaris	Connection library for Windows
Sybase Client Library 11.0	libjxa.so	libjxa.dll	libjct_r.so	libjct.dll
Oracle OCI 7.x	libclntsh.so	xa73.dll	libclntsh.so	ociw32.dll
Oracle OCI 8.x	libclntsh.so	oraclient8.dll	libclntsh.so	oci.dll
Oracle OCI 9.x	libclntsh.so	oraclient9.dll	libclntsh.so	oci.dll

Note In most cases, you should not need to modify the resource manager properties.

However, to use a shared library or DLL other than the default, you must edit the database property file. For example, for XA resource connections using Oracle OCI 8.*x*, where 8.*x* is 8.1.5 or lower, set this property value in the *Repository/ResourceManager/OCI_8.props* file:

```
com.sybase.jaguar.resourcemanager.xalib = xa80.dll
```

See also Chapter 4, “Database Access”

com.sybase.jaguar.resourcemanager.check

Description Specifies the query to use when testing whether a connection is still usable.

Syntax The SQL query text. The default is:

```
select 1
```

The default does not work on all databases. For example, for an Oracle database, you must set this property to:

```
select 1 from dual
```

See also com.sybase.jaguar.resourcemanager.checkallowed

com.sybase.jaguar.resourcemanager.checkallowed

Description	Specifies whether the cache manager should test connections before returning them to the cache.
Syntax	`true` or `false`. The default is `true`.
Usage	The query used to test the cache is specified by the com.sybase.jaguar.resourcemanager.check property.

com.sybase.jaguar.resourcemanager.conlib

Description	Specifies the connection library used for this connection type.
Syntax	The syntax is the same as the DLL or Class Name field on the Driver tab in the Connection Cache Properties dialog box. See “Driver properties” on page 84.

com.sybase.jaguar.resourcemanager.conlib.aix

Description	Specifies the connection library used for this connection type on AIX.
Syntax	The syntax is the same as the DLL or Class Name field on the Driver tab in the Connection Cache Properties dialog box. See “Driver properties” on page 84.

com.sybase.jaguar.resourcemanager.conlib.hpux

Description	Specifies the connection library used for this connection type on HP-UX.
Syntax	The syntax is the same as the DLL or Class Name field on the Driver tab in the Connection Cache Properties dialog box. See “Driver properties” on page 84.

com.sybase.jaguar.resourcemanager.conlib.linux

Description	Specifies the connection library used for this connection type on Linux.
Syntax	The syntax is the same as the DLL or Class Name field on the Driver tab in the Connection Cache Properties dialog box. See “Driver properties” on page 84.

com.sybase.jaguar.resourcemanager.conlib.nt

Description — Specifies the connection library (DLL) used for this connection type on Windows 2000 or Windows XP.

Syntax — The syntax is the same as the DLL or Class Name field on the Driver tab in the Connection Cache Properties dialog box. See “Driver properties” on page 84.

com.sybase.jaguar.resourcemanager.conlib.solaris

Description — Specifies the connection library used for this connection type on Solaris.

Syntax — The syntax is the same as the DLL or Class Name field on the Driver tab in the Connection Cache Properties dialog box. See “Driver properties” on page 84.

com.sybase.jaguar.resourcemanager.description

Description — Specifies an optional text description of the XA resource.

Syntax — *desc*

Where *desc* is the descriptive text.

com.sybase.jaguar.resourcemanager.enabled

Description — Specifies whether connections for this connection type are enabled.

Syntax — `true` or `false`. The default is `true`.

Usage — For debugging, you may want to set this property to `false`, which disables all connections using this connection type. For example, to disable all OCI_8 connection caches, set this property to `false` in the *OCI_8.props* file.

com.sybase.jaguar.resourcemanager.name

Description — Specifies the name of the connection type.

Syntax — A text string that represents the library type. In each of these database property files, the names are:

Database property file	Name	To indicate
CTLIB_110.props	CTLIB_110	Sybase Open Client Client-Library connections
OCI_7.props	OCI_7	Connections using OCI 7.x
OCI_8.props	OCI_8	Connections using OCI 8.x
OCI_9.props	OCI_9	Connections using OCI 9.x

com.sybase.jaguar.resourcemanager.ssa

Description — Enables set-proxy support for connections to databases that support this feature.

Syntax — `true` or `false`. The default is `false`, which disables set-proxy support.

Usage — Current versions of Adaptive Server Enterprise allow a user to assume the identity and privileges of another user. You can use this feature only with Sybase Open Client Client-Library connections, using this command:

```
set session authorization "login-name"
```

When proxy support is enabled, connections retrieved from the connection cache are set to act as a proxy for the user name associated with the EAServer client. To set proxy to another user name, use the Java JCMCache.getProxyConnection() method or the C JagCmGetProxyConnection() routine in your component.

com.sybase.jaguar.resourcemanager.type

Description — Specifies the connection library type for this connection.

Syntax — A text string that represents the library type. In each of these database property files, the types are:

Database property file	Type
CTLIB_110.props	CTLIB_110
OCI_7.props	OCI_7
OCI_8.props	OCI_8
OCI_9.props	OCI_9

com.sybase.jaguar.resourcemanager.xacompliant

Description — Specifies whether this connection library type supports XA resource connections.

Syntax — `true` or `false`. The default is `true` for the connection library types listed in Table B-17 on page 490.

com.sybase.jaguar.resourcemanager.xalib

Description — Specifies the XA-Library used for this connection.

Syntax — The syntax is the same as the Connection Library XA field on the Driver tab of the Connection Cache Properties dialog box. See Table 4-3 on page 87.

com.sybase.jaguar.resourcemanager.xalib.aix

Description — Specifies the XA-Library used for this connection type on AIX.

Syntax — The syntax is the same as the Connection Library XA field on the Driver tab of the Connection Cache Properties dialog box. See Table 4-3 on page 87.

com.sybase.jaguar.resourcemanager.xalib.hpux

Description — Specifies the XA-Library used for this connection type on HP-UX.

Syntax — The syntax is the same as the Connection Library XA field on the Driver tab of the Connection Cache Properties dialog box. See Table 4-3 on page 87.

com.sybase.jaguar.resourcemanager.xalib.linux

Description — Specifies the XA-Library used for this connection type on Linux.

Syntax — The syntax is the same as the Connection Library XA field on the Driver tab of the Connection Cache Properties dialog box. See Table 4-3 on page 87.

com.sybase.jaguar.resourcemanager.xalib.nt

Description — Specifies the XA-Library (DLL) used for this connection type on Windows 2000 or Windows XP.

Syntax — The syntax is the same as the Connection Library XA field on the Driver tab of the Connection Cache Properties dialog box. See Table 4-3 on page 87.

com.sybase.jaguar.resourcemanager.xalib.solaris

Description — Specifies the XA-Library used for this connection type on Solaris.

Syntax — The syntax is the same as the Connection Library XA field on the Driver tab of the Connection Cache Properties dialog box. See Table 4-3 on page 87.

Resource reference properties

Description — For application clients, EJB components, and Web applications, you can define resource reference properties. Resource references specify aliased JNDI names for database connections, JavaMail sessions, and URL factories used by the application code.

Resource reference syntax — Resource reference properties are of the form:

res-ref1, *res-ref2*, ...

Where *res-ref1*, *res-ref2*, and so forth are of the form:

(description=*desc*,res-type=*type*,res-auth=*auth*,
res-sharing-scope=*sharing*,res-link=*link*,
res-ref-name=*jndi-name*)

Where:

- *desc* is an optional comment describing how the entry is to be set.
- *type* is the Java datatype of the reference, which is one of:
  - java.net.URL
  - javax.mail.Session
  - javax.sql.Datasource
- *auth* is “Container” or “Application”.
- *sharing* is “Shareable” or “Unshareable”.

- *link* is:
  - For javax.sql.Datasource references, the name of the connection cache. The cache must be defined and allow access by name; in other words, the com.sybase.jaguar.conncache.cachebyname must be true.
  - For javax.mail.Session references, the SMTP mail server host name.
  - For java.net.URL references, the URL.
- *jndi-name* is entry's JNDI name, relative to the *java:comp/env* prefix.

See also com.sybase.jaguar.applicationclient.resource-ref, com.sybase.jaguar.component.resource-ref, com.sybase.jaguar.webapplication.resource-ref

Role properties

Description Role property names begin with com.sybase.jaguar.role. Roles can be associated with packages, Web applications, components, and methods to constrain which users can access a resource. In EAServer Manager, configure roles using the Roles folder.

com.sybase.jaguar.role.authorizeddigitalids

Description Specifies digital certificate IDs that can assume this role.

Syntax A comma-separated list of digital ID names. Each must correspond to the common name of a certificate that is installed in the repository certificate database.

Usage In EAServer Manager, set this property using the role's Authorized Digital ID folder.

See also com.sybase.jaguar.role.excludeddigitalids

com.sybase.jaguar.role.authorizedgroups

Description Specifies operating system group names that are part of this role.

Syntax A comma-separated list of group names.

Usage	In EAServer Manager, set this property with the role's Authorized Groups folder.
See also	com.sybase.jaguar.role.excludedgroups

com.sybase.jaguar.role.authorizedusers

Description	Specifies user IDs that can assume this role.
Syntax	A comma-separated list of user IDs.
Usage	In EAServer Manager, set this property using the role's Authorized Digital ID folder.
See also	com.sybase.jaguar.role.excludeddigitalids

com.sybase.jaguar.role.description

Description	Specifies the description of this role.
Syntax	*desc* Where *desc* is the descriptive text.
Usage	In EAServer Manager, set this property using the Description field on the General tab of the Role Properties dialog box.

com.sybase.jaguar.role.excludeddigitalids

Description	Specifies digital certificate IDs that cannot assume this role.
Syntax	A comma-separated list of digital ID names. Each must correspond to the common name of a certificate that is installed in the repository certificate database.
Usage	In EAServer Manager, set this property using the role's Authorized Digital ID folder.
See also	com.sybase.jaguar.role.authorizeddigitalids

com.sybase.jaguar.role.excludedgroups

Description	Specifies operating system group names that are excluded from this role.

Syntax	A comma-separated list of group names.
Usage	In EAServer Manager, set this property with the role's Authorized Groups folder.
See also	com.sybase.jaguar.role.authorizedgroups

com.sybase.jaguar.role.excludedusers

Description	Specifies digital certificate IDs that cannot assume this role.
Syntax	A comma-separated list of digital ID names. Each must correspond to the common name of a certificate that is installed in the repository certificate database.
Usage	In EAServer Manager, set this property using the role's Authorized Digital ID folder.
See also	com.sybase.jaguar.role.authorizeddigitalids

com.sybase.jaguar.role.name

Description	The name of this role.
Syntax	*role-name* Where *role-name* is the application name.
Usage	In EAServer Manager, the role name is set when creating the role and cannot be changed.

com.sybase.jaguar.role.roleowner

Description	Specifies the user that owns this role.
Syntax	*user* Where *user* is a user name. The default is jagadmin.
Usage	Only the role owner or a member of the Administrative role can modify a role's properties. In EAServer Manager, set this property in the Owner field in the Role Properties dialog box.

Security properties

Description

Security property names begin with `com.sybase.jaguar.security`. Security entities can represent security profiles, used to configure SSL settings for server listeners, and security identities, used for intercomponent calls and interserver authentication. The com.sybase.jaguar.security.type property determines what type the entity is.

Not all security properties apply to both identities and profiles. If a property applies only to one or the other, the description will say so.

com.sybase.jaguar.description

Description

Specifies a text description of the identity or profile.

Syntax

desc

Where *desc* is the descriptive text.

com.sybase.jaguar.security.cachetime

Description

For profiles, specifies the time in seconds to cache SSL session parameters. Not used for identities.

Syntax

An integer value representing the cache timeout in seconds. If not set, the default is 28800 (which equals 8 hours).

See also

com.sybase.jaguar.security.sesscachesize

com.sybase.jaguar.security.certname

Description

Specifies the name of the SSL certificate. This property is always required for security profiles. For identities, this property is required if the com.sybase.jaguar.security.qoss property specifies a security characteristic that requires mutual SSL authentication.

Syntax

The text of the certificate label, for example:

```
Sample1 Test ID
```

See also

com.sybase.jaguar.security.qoss

com.sybase.jaguar.security.entrustinifile

Description — For profiles or identities that use Entrust PKI software to manage certificates, specifies the full path to the Entrust INI file that provides information on how to access Entrust.

Syntax — The full path to the file, for example on Windows platforms:

```
c:\program files\entrust\entrust.ini
```

Or on a UNIX platform:

```
/opt/Entrust/clients/entrust.ini
```

The actual path depends on where you or your system administrator have installed the Entrust software.

Usage — This property must be set when the com.sybase.jaguar.security.tokentype property is set to “entrust”.

com.sybase.jaguar.security.entrustpassword

Description — For profiles or identities that use Entrust PKI software to manage certificates, specifies the password for access to the Entrust user profile.

Syntax — The password text. Values are encrypted in the repository.

Usage — This property must be set when the com.sybase.jaguar.security.tokentype property is set to “entrust”.

See also — com.sybase.jaguar.security.entrustuserprofile

com.sybase.jaguar.security.entrustuserprofile

Description — For profiles or identities that use Entrust PKI software to manage certificates, specifies the full path to the Entrust user profile, which is an operating system file.

Syntax — The full path to the file, for example on Windows platforms:

```
c:\my documents\entrust\CN=Sample Entrust User, O=Sybase, C=US.epf
```

Or on a UNIX platform:

```
/opt/Entrust/certs/CN=Sample Entrust User, O=Sybase, C=US.epf
```

The actual path depends on where you or your system administrator have created the Entrust profile.

Usage	This property must be set when the com.sybase.jaguar.security.tokentype property is set to “entrust”.

com.sybase.jaguar.security.logpeerIP

Description	For security profiles, specifies whether to log the client IP address for SSL connection failures associated with the security profile.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.security.logsslerr

com.sybase.jaguar.security.logsslerr

Description	For security profiles, specifies whether to log additional SSL error information for connections associated with the security profile.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.listener.logsslerr

com.sybase.jaguar.security.name

Description	Specifies the profile or identity name.
Syntax	*app-name* Where *app-name* is the application name.

com.sybase.jaguar.security.passphrase

Description	For identities or profiles, specifies the password to access the Sybase certificate database.
Syntax	The password text. Values are encrypted in the repository.

com.sybase.jaguar.security.qoss

Description — For identities or profiles, specifies the name of the security characteristic to use. For profiles, the security characteristic determines the minimum level of security acceptable for an incoming connection. For identities, the security characteristic determines the minimum level of security acceptable for outgoing connections.

Syntax — If a value is specified, it must match the name of the security profile. The *EAServer Security Administration and Programming Guide* describes the available security profile names. The list of available profiles can be retrieved programmatically using the CtsSecurity::SSLServiceProvider interface.

For identities, if this property is not set, or set with no value, outgoing connections do not use SSL. For profiles, this property must be set to the name of a security characteristic.

com.sybase.jaguar.security.sesscachesize

Description — For profiles used by a listener, specifies the size of SSL session cache. Not used for identities.

Syntax — An integer that specifies the number of SSL sessions to cache. If no value is specified, the default is the value of the com.sybase.jaguar.server.maxconnections server property. When you save profile information from EAServer Manager, the default is 30.

Usage — EAServer caches server-side SSL sessions to improve performance when clients create frequent short-lived secure connections, typically from Web browsers. Caching improves performance by eliminating the time required to recreate sessions for the same client.When a security session is reused, clients avoid a CPU-intensive encryption of the premaster-secret using the server's public key. Similarly, servers avoid a CPU-intensive decryption of the premaster-secret using its private key. The client must send the SSL session ID from the previous connection for the session to be reused.

For best performance, set the cache size to a number less than or equal to the com.sybase.jaguar.server.maxconnections server property. The cache requires approximately 64 bytes per entry.

See also — com.sybase.jaguar.security.cachetime, com.sybase.jaguar.security.sessshare

com.sybase.jaguar.security.sessshare

Description	For profiles, specifies the maximum number of concurrent SSL sessions that can share the same session parameters. Not used for identities.
Syntax	An integer number. If no value is specified, the default is 10.
Usage	SSL session sharing allows a client to use the same SSL session ID for multiple connections. Session sharing can improve performance when the client opens multiple connections simultaneously. For example, a browser client may open several connections to download images linked to an HTML page. Session sharing allows the client to reuse the session for the second and subsequent connections, up to the number of concurrent connections specified by this property.
See also	com.sybase.jaguar.security.sesscachesize, com.sybase.jaguar.security.cachetime

com.sybase.jaguar.security.specifiedidentity

Description	For an identity, specifies the user name to be used for component invocations or outgoing interserver connections.
Syntax	The user name.
See also	com.sybase.jaguar.security.specifiedidentitypassphrase

com.sybase.jaguar.security.specifiedidentitypassphrase

Description	For an identity, specifies the password to be used for component invocations or outgoing interserver connections.
Syntax	The password text. Values are encrypted in the repository.
See also	com.sybase.jaguar.security.specifiedidentity

com.sybase.jaguar.security.tokentype

Description	Specifies whether to use the Sybase SSL certificate database or an Entrust certificate.
Syntax	Allowable values are `sybase` and `entrust`.

com.sybase.jaguar.security.type

Description — Specifies the security entity type.

Syntax — Allowable values are:

Table B-19: Security entity type values

Value	To indicate
identity	An identity
listener	A security profile

Server properties

Description — Server property names begin with `com.sybase.jaguar.server`. A server represents an application server process.

See also — Chapter 3, "Creating and Configuring Servers"

com.sybase.jaguar.server.apichk

Description — Specifies whether API checking is enabled for Open Server Server-Library calls.

Syntax — `true` or `false`. The default is `true`.

Usage — This property affects only code that calls Open Server Server-Library routines. See Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide* for information on running Open Server applications in EAServer.

com.sybase.jaguar.server.applications

Description — Specifies the applications that are installed in this server.

Syntax — A comma-separated list of application names.

See also — com.sybase.jaguar.server.packages, com.sybase.jaguar.server.webapplications, Application properties on page 345

com.sybase.jaguar.server.authentication

Description	Specifies whether operating-system-based user name/password authentication is in effect.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.server.authservice, com.sybase.jaguar.server.jaas.config, com.sybase.jaguar.server.jagadminpassword

com.sybase.jaguar.server.authlockout

Description	Specifies how long a user account is locked after five failed login attempts.
Syntax	An integer specifying the lockout period, in seconds. The default is 600 (10 minutes).

com.sybase.jaguar.server.authorization.permcachetimeout

Description	The length of time, in seconds, that the server can cache authorization data for a user's access to a resource.
Syntax	An integer specifying the number of seconds; the default is 7200, which is 2 hours.

com.sybase.jaguar.server.authorization.service

Description	Specifies the name of a custom authorization service component.
Syntax	A URL to identify the custom authorization service component.

There are two accepted forms of the URL:

- For all component types, the URL can be set to the *EAServerPackage/EAServerComponent*; the component must be installed in the server.
- Java CORBA and C++ CORBA components can be accessed using the pseudocomponent object URL. The syntax for a Java pseudocomponent is:

```
pseudo://java/JavaClass/package/component
```

Where *JavaClass* is the Java class name, *package* is the EAServer package name, and *component* is the component name. The syntax for a C++ pseudocomponent is:

```
pseudo://cpp/SharedLibraryName/package/component
```

Where *SharedLibraryName* is the DLL or UNIX shared library file name.

Components implemented for pseudocomponent access must be thread-safe, and you must restart EAServer to refresh the component.

For more information on pseudocomponents, see Chapter 34, "Creating and Using EAServer Pseudocomponents," in the *EAServer Programmer's Guide*.

Usage

The *EAServer Security Administration and Programming Guide* explains how to create a custom authorization service.

See also

com.sybase.jaguar.server.authentication,
com.sybase.jaguar.server.authservice,
com.sybase.jaguar.server.jaas.config,
com.sybase.jaguar.server.roleservice

com.sybase.jaguar.server.authservice

Description

Specifies the name of a custom authentication service component.

Syntax

The authentication service component name, in the form:

package/component

The package must be installed on the server, and the component must implement the interface CtsSecurity::AuthService.

Usage

The *EAServer Security Administration and Programming Guide* explains how to create a custom authentication service.

See also

com.sybase.jaguar.server.authentication,
com.sybase.jaguar.server.authorization.service,
com.sybase.jaguar.server.jaas.config,
com.sybase.jaguar.server.roleservice

com.sybase.jaguar.server.authtimeout

Description

Specifies how long authentication results can be cached.

Syntax

The cache timeout period in seconds. The default is 3600 (1 hour).

See also com.sybase.jaguar.server.authentication, com.sybase.jaguar.server.authservice

com.sybase.jaguar.server.bindrefresh

Description Specifies when the name service binds component names.

Syntax

Value	To indicate
`run` (the default)	Components are bound in the name service run method.
`start`	Components are bound in the name service start method. Use this value if you have service components that make intercomponent calls in their start method.

com.sybase.jaguar.server.bindservers

Description Specifies other servers that use this server for naming services.

Syntax This property is set automatically as servers bind to the name service, and should not be set manually.

See also com.sybase.jaguar.server.CosNaming.nameserver,
Chapter 5, “Naming Services”

com.sybase.jaguar.server.classloader.debug

Description Enables custom class loader tracing.

Syntax `true` or `false`. The default is `false`, which disables class loader tracing.

Usage This property is useful for determining which custom loader (server, application, package, Web application, or component) is loading a particular class for a particular component or Web application.

In EAServer Manager, set this property using the Advanced tab in the Server Properties dialog box.

See also com.sybase.jaguar.server.java.classes,
com.sybase.jaguar.server.jvm.verbose,
com.sybase.jaguar.server.jvm.verboseGC,
com.sybase.jaguar.component.java.classes,
com.sybase.jaguar.application.java.classes,
com.sybase.jaguar.package.java.classes,
com.sybase.jaguar.webapplication.java.classes

Chapter 30, “Configuring Custom Java Class Lists,” in the *EAServer Programmer’s Guide*

com.sybase.jaguar.server.cluster.IPV6serverID

Description — For servers that run in clusters and support Internet Protocol Version 6 (IPV6), specifies a server ID that is unique in the cluster.

Syntax — A string in the format of an IPV4 address that is unique in the cluster, for example:

```
10.123.456.789
```

If you do not specify a value, EAServer generates a server ID, however, the generated ID is not guaranteed to be unique in the cluster if servers use IPV6 addresses.

See also — “Starting the server” on page 49.

com.sybase.jaguar.server.cmp_iso_level

Description — Specifies a default effective transaction isolation level for EJB CMP entity beans installed in the server.

Syntax — See “Configuring CMP isolation level” in the *EAServer Performance and Tuning Guide*.

See also — com.sybase.jaguar.component.cmp_iso_level

com.sybase.jaguar.server.code.set

Description — Specifies the default for the com.sybase.jaguar.package.code.set package property.

Syntax — Same as com.sybase.jaguar.component.code.set. The default is utf8.

Usage — In EAServer Manager, set this property using the Codeset field on the General tab in the Server Properties dialog box.

See also — com.sybase.jaguar.component.code.set, com.sybase.jaguar.package.code.set

com.sybase.jaguar.server.CosNaming.bindpassword

Description	Specifies the password to bind to the naming service.
Syntax	The password text.
See also	“Name binding password security” on page 119.

com.sybase.jaguar.server.CosNaming.heartbeat

Description	Enables heart beat detection for the name service.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.server.CosNaming.heartbeatfrequency, “Heartbeat detection” on page 129

com.sybase.jaguar.server.CosNaming.heartbeatfrequency

Description	When com.sybase.jaguar.server.CosNaming.heartbeat is `true`, specifies the heartbeat frequency.
Syntax	An integer that specifies the number of seconds between heartbeat tests. The default is 120.
See also	com.sybase.jaguar.server.CosNaming.heartbeat, “Heartbeat detection” on page 129

com.sybase.jaguar.server.CosNaming.initialcontext

Description	Specifies the initial naming context.
Syntax	The initial naming context. The default is a zero-length string.
See also	Chapter 5, “Naming Services”

com.sybase.jaguar.server.CosNaming.ldapurl

Description	When com.sybase.jaguar.server.CosNaming.strategy is `persistent`, specifies the URL to connect to the LDAP server.
Syntax	The URL text.

See also com.sybase.jaguar.server.CosNaming.strategy,
"Using an LDAP server with EAServer" on page 119

com.sybase.jaguar.server.CosNaming.mgrdn

Description When com.sybase.jaguar.server.CosNaming.strategy is `persistent`, specifies the user name to connect to the LDAP server.

Syntax The user name.

See also com.sybase.jaguar.server.CosNaming.strategy,
com.sybase.jaguar.server.CosNaming.mgrpwd,
"Using an LDAP server with EAServer" on page 119

com.sybase.jaguar.server.CosNaming.mgrpwd

Description When com.sybase.jaguar.server.CosNaming.strategy is `persistent`, specifies the password to connect to the LDAP server.

Syntax The password text.

See also com.sybase.jaguar.server.CosNaming.strategy,
com.sybase.jaguar.server.CosNaming.mgrdn,
"Using an LDAP server with EAServer" on page 119

com.sybase.jaguar.server.CosNaming.nameserver

Description Specifies whether this server acts as its own name server, or connects to another EAServer engine for naming services.

Syntax `true` or `false`. The default is `true`.

See also com.sybase.jaguar.server.nameservice,
Chapter 5, "Naming Services"

com.sybase.jaguar.server.CosNaming.strategy

Description When com.sybase.jaguar.server.CosNaming.nameserver is `true`, specifies whether to store naming services internally in server memory, or externally in a name server.

Syntax

Value	To indicate
`transient` (the default)	Name bindings are stored internally in server memory.
`persistent`	Name bindings are stored externally in an LDAP name server.

See also "Transient versus persistent storage" on page 107

com.sybase.jaguar.server.debug.useagent

Description Specifies the interface supported for remote Java debugging.

Syntax

Value	To indicate
`true` (the default)	Java debugging supported by the sun.tools.debug interface.
`false`	Java debugging is supported by the JPDA interface, at the port specified by the com.sybase.jaguar.server.jpda.port property.

See also com.sybase.jaguar.server.jpda.port

com.sybase.jaguar.server.defaultStorageCache

Description Specifies the default connection cache for EJB CMP entity beans that are imported from an EJB-JAR file.

Syntax The name of an existing JDBC connection cache. If not set, the default is `JavaCache`.

Usage The connection cache for EJB CMP entity beans is set as `cache` part of the com.sybase.jaguar.component.storage component property. To override the default, you can set this property with the *sybase-easerver-config.xml* configuration file in the EJB-JAR file.

See also com.sybase.jaguar.component.storage,
"Using EAServer configuration files in J2EE archives" on page 196,
Chapter 27, "Creating Entity Components," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.description

Description	Specifies an optional text description of the server.
Syntax	The descriptive text.

com.sybase.jaguar.server.destroyPooledInstancesOnShutdown

Description	Whether to destroy pooled component instances when shutting down the server.
Syntax	A value of `true` indicates that pooled instances must be destroyed explicitly before shutting down. For example, an EJB component's ejbRemove method will be called. This allows the pooled instance to clean up resources, such as closing database connections. A value of `false` specifies that instances are not destroyed. The default is `true`.
Usage	If many component instances are pooled, explicit destruction of instances may lengthen the time required to shut down or restart the server. The server setting can be overridden for individual components by setting the com.sybase.jaguar.component.destroyPooledInstancesOnShutdown component property.
See also	com.sybase.jaguar.server.destroyPooledInstancesOnShutdownTimeout, com.sybase.jaguar.component.destroyPooledInstancesOnShutdown

com.sybase.jaguar.server.destroyPooledInstancesOnShutdownTimeout

Description	How long to wait for each instance destruction method to return when destroying pooled instances during server shutdown.
Syntax	The time to wait, in seconds. If no value is specified, the default is 5.
See also	com.sybase.jaguar.server.destroyPooledInstancesOnShutdown, com.sybase.jaguar.component.destroyPooledInstancesOnShutdownTimeout

com.sybase.jaguar.server.disconnect

Description	Specifies whether or not the Open Server ATTENTION event handler is called when a client disconnects.
Syntax	`true` or `false`. The default is `false`.

See also	com.sybase.jaguar.server.handler.attnevent, com.sybase.jaguar.server.handler.disconnevent

com.sybase.jaguar.server.DOMfactory

Description	Specifies the class name for a custom DOM XML parser factory class.
Syntax	The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.server.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.
See also	com.sybase.jaguar.server.jagmgr.DOMFactoryChoice, com.sybase.jaguar.server.SAXfactory, com.sybase.jaguar.server.XSLTfactory

com.sybase.jaguar.server.dynamo.exec

Description	Enables and disables PowerDynamo Web site execution.
Syntax	`true` or `false`. The default is `false`.

com.sybase.jaguar.server.dynamo.shlib

Description	Specifies the name of the PowerDynamo NSAPI plug-in.
Syntax	The library or DLL name.

com.sybase.jaguar.server.dynamo.trace

Description	Enables and disables tracing of PowerDynamo plug-in calls.
Syntax	`true` or `false`. The default is `false`.

com.sybase.jaguar.server.ejb.role.default

Description	Specifies the default required role for EJB component methods that do not have a method-level role setting.

Syntax

Value	To indicate
`nobody` (the default)	No user can execute the method.
`everybody`	All users can execute the method.

Usage

The default value for this property provides compliance with the EJB 2.0 specification.

Behavior in EAServer 4.0 versus 4.1 and later The effect of the com.sybase.jaguar.server.ejb.role.default property on EJB 2.0 components in EAServer 4.0 differs from EAServer version 4.1 and later:

- The com.sybase.jaguar.server.ejb.role.default property is a server-wide property. To assign roles to all methods of an EJB 2.0 component in EAServer 4.0, you can assign individual roles to all methods of the component, or set the com.sybase.jaguar.server.ejb.role.default property to everybody.
- Beginning with EAServer 4.1, com.sybase.jaguar.server.ejb.role.default has been removed. To assign a role to any one method of an EJB 2.0 component, assign roles to all methods of the component. Otherwise, the server displays an error message when trying to execute the method that does not have a role assigned to it.

  If none of the methods of an EJB 2.0 component have roles assigned to them, authorization checks are not enforced and authorization is not performed. No error message is sent to the server log.

See also

com.sybase.jaguar.method.security-roles

com.sybase.jaguar.server.external.request.timeout

Description

For components that are installed on this server but run externally, specifies how long to wait for a response from the external server before returning an error to the client.

Syntax

The timeout in seconds. If not set, the default is 60 seconds. A value of 0 specifies infinity.

See also

com.sybase.jaguar.component.external.request.timeout

com.sybase.jaguar.server.external.serverstart.timeout

Description	For components that are installed on this server but run externally, specifies how long to wait for a response from the external server before returning an error to the client.
Syntax	The timeout in seconds. If not set, the default is 60 seconds. A value of 0 specifies infinity.
See also	com.sybase.jaguar.component.external.serverstart.timeout

com.sybase.jaguar.server.filter-mapping

Description	Associates a custom response header filter with URL paths. The default mapping is "/*"; this runs the filter on all server resources.
Syntax	*mapping1*, *mapping2*, *mapping3*, ... Where *mapping1*, *mapping2*, *mapping3* are strings of the form: (description=*desc*,filter-name=*filter*, type=*pattern*) Where: • *desc* is an optional description of the mapping. • *filter* is the filter name. • *pattern* is the URL pattern.
See also	"HTTP Custom Response Header" on page 41

com.sybase.jaguar.server.filter.init-param

Description	Specifies a string of name/value pairs used to define server-level HTTP custom headers.
Syntax	A string of the form: *entry1*, *entry2*, ... Where *entry1*, *entry2*, and so forth are of the form: (description=,value=type:String&val:*value,*name=*name*) Where *value* is the property value and *name* is the property name.
See also	"HTTP Custom Response Header" on page 41

com.sybase.jaguar.server.flowcontrol.http

Description	Enables flow control for HTTP client requests.
Syntax	`true` or `false`. The default is `false`, which disables flow control.
Usage	When the server is very busy with many client connections, client request threads may repeatedly conflict with each other for access to low-level system resources. Flow control provides a coarser level of granularity for synchronizing access to system resources by request threads. When enabled, flow control can improve performance by replacing multiple, serial choke points in the request processing sequence with a single choke point.
See also	com.sybase.jaguar.server.flowcontrol.iiop, com.sybase.jaguar.server.flowcontrol.maxexethreads

com.sybase.jaguar.server.flowcontrol.iiop

Description	Enables flow control for IIOP client requests.
Syntax	true or false. The default is false, which disables flow control.

Warning! In some scenarios, IIOP client threads may deadlock when IIOP flow control is enabled. Do not enable IIOP flow control in production servers without first stress testing your applications with scenarios that reflect production loads.

See also	com.sybase.jaguar.server.flowcontrol.http, com.sybase.jaguar.server.flowcontrol.maxexethreads

com.sybase.jaguar.server.flowcontrol.maxexethreads

Description	Specifies the maximum number of threads that can concurrently execute code that is governed by flow control.
Syntax	Specify a positive integer. If not set, the default is the value of the com.sybase.jaguar.server.maxthreads property, which means all threads can proceed. For the best performance, you should tune this number to get the best performance under peak stress conditions. Values between 15 and 30 are a good starting point. To tune the setting, monitor response time at peak load conditions, and raise or lower the value to find the setting that results in the best response time.

See also com.sybase.jaguar.server.flowcontrol.http,
com.sybase.jaguar.server.flowcontrol.iiop,
com.sybase.jaguar.server.maxthreads

com.sybase.jaguar.server.handler.attnevent

Description Specifies an Open Server ATTENTION event handler.

Syntax *library:function*

Where *library* is the DLL or shared library name, and *function* is the function name.

See also Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.bulkevent

Description Specifies an Open Server BULK event handler.

Syntax *library:function*

Where *library* is the DLL or shared library name, and *function* is the function name.

See also Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.connevent

Description Specifies an Open Server CONNECT event handler.

Syntax *library:function*

Where *library* is the DLL or shared library name, and *function* is the function name.

See also Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.crsevent

Description Specifies an Open Server CURSOR event handler.

Syntax	*library*:*function* Where *library* is the DLL or shared library name, and *function* is the function name.
See also	Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.disconnevent

Description	Specifies an Open Server DISCONNECT event handler.
Syntax	*library*:*function* Where *library* is the DLL or shared library name, and *function* is the function name.
See also	Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.dynamicevent

Description	Specifies an Open Server DYNAMIC event.
Syntax	*library*:*function* Where *library* is the DLL or shared library name, and *function* is the function name.
See also	Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.errorevent

Description	Specifies an Open Server ERROR event handler.
Syntax	*library*:*function* Where *library* is the DLL or shared library name, and *function* is the function name.
See also	Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.initevent

Description	Specifies an Open Server INITIALIZATION event handler.
Syntax	*library:function* Where *library* is the DLL or shared library name, and *function* is the function name.
See also	Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.langevent

Description	Specifies an Open Server LANGUAGE event handler.
Syntax	*library:function* Where *library* is the DLL or shared library name, and *function* is the function name.
See also	Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.msgevent

Description	Specifies an Open Server MESSAGE event handler.
Syntax	*library:function* Where *library* is the DLL or shared library name, and *function* is the function name.
See also	Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.optionevent

Description	Specifies an Open Server OPTIONS event handler.
Syntax	*library:function* Where *library* is the DLL or shared library name, and *function* is the function name.

See also — Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.rpcevent

Description — Specifies an Open Server RPC event handler.

Syntax — *library*:*function*

Where *library* is the DLL or shared library name, and *function* is the function name.

See also — Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.startevent

Description — Specifies an Open Server START event handler.

Syntax — *library*:*function*

Where *library* is the DLL or shared library name, and *function* is the function name.

See also — Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.handler.stopevent

Description — Specifies an Open Server STOP event handler.

Syntax — *library*:*function*

Where *library* is the DLL or shared library name, and *function* is the function name.

See also — Appendix B, "Migrating Open Server Applications to EAServer," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.hotstandby

Description — Specifies whether this server is in a hot standby pair.

Syntax	`true` or `false`. The default is `false`.
See also	"Hot Standby" on page 37, com.sybase.jaguar.server.hotstandby.backup, com.sybase.jaguar.server.hotstandby.master

com.sybase.jaguar.server.hotstandby.backup

Description	When com.sybase.jaguar.server.hotstandby is `true`, specifies the backup server URL in the hot standby pair.
Syntax	The IIOP or IIOPS URL of the backup server.
See also	"Hot Standby" on page 37, com.sybase.jaguar.server.hotstandby, com.sybase.jaguar.server.hotstandby.master

com.sybase.jaguar.server.hotstandby.master

Description	When com.sybase.jaguar.server.hotstandby is `true`, specifies the master server URL in the hot standby pair.
Syntax	The IIOP or IIOPS URL of the master server.
See also	"Hot Standby" on page 37, com.sybase.jaguar.server.hotstandby, com.sybase.jaguar.server.hotstandby.backup

com.sybase.jaguar.server.http.acceptlang

Description	Enables and disables HTTP accept-language header parsing.
Syntax	`true` or `false`. The default is `false`.

com.sybase.jaguar.server.http.cache.debug

Description	Specifies whether to write static page cache debug information to the server's log file.
Syntax	`true` or `false`. The default is `false`.

com.sybase.jaguar.server.http.cache.enable

Description	Specifies whether to enable static page caching.
Syntax	`true` or `false`. The default is `true`.
See also	"Static Page Caching" on page 39

com.sybase.jaguar.server.http.cache.exclude-files

Description	This property is no longer supported. To exclude files from the cache, use the com.sybase.jaguar.server.http.cache.webapps.exclude-files property.
Syntax	The value is ignored.

com.sybase.jaguar.server.http.cache.size

Description	Specifies the size of the static page cache.
Syntax	An integer followed by "B", "K", or "M", indicating bytes, kilobytes, and megabytes, respectively. The default is 10M. You can specify the size using either uppercase or lowercase letters; for example, to set the cache size to 20 megabytes, you can enter `20M` or `20m`.
See also	"Static Page Caching" on page 39, com.sybase.jaguar.server.http.cache.timeout

com.sybase.jaguar.server.http.cache.timeout

Description	Specifies the timeout period for entries in the static page cache. When an entry in the cache times out, it becomes invalid. The page is removed from the cache the next time it is requested or when space is needed for a new cache entry.
Syntax	The timeout period in seconds. The default is 600 (10 minutes).
See also	"Static Page Caching" on page 39

com.sybase.jaguar.server.http.cache.webapps.exclude-files

Description	When static page caching is enabled, specifies the Web application files to exclude from caching.

Syntax

A comma-delimited string that specifies the Web application files to exclude from the cache. Enter the string in this form; items in brackets are optional:

(<*WebAppName* >[/<*dir*>], [<*file type*>], [<*file type*>], ...),
(<*WebAppName*>[/<*dir*>], [<*file type*>], ...), ...

For example, to exclude all the GIF and JPG files in the *images* directory and all the files in the *archives* directory for the Web application "Vacation", enter:

```
(Vacation/images, *.gif, *.jpg),
(Vacation/archives, *.*)
```

See also

"Static Page Caching" on page 39,
com.sybase.jaguar.server.http.cache.enable

com.sybase.jaguar.server.http.defaultwebapp

Description

Specifies the Web application to invoke when users access the default URL http://*host*:*port*/. If not set, the welcome page of the Web application with the context path "/" is invoked; by default, this is the EASDefault Web application, and its welcome page is *$JAGUAR/html/index.html*.

Syntax

The name of a Web application.

See also

"The EASDefault Web application" in Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide.*

com.sybase.jaguar.server.http.dirbrowseenable

Description

Enables browsing of directories for HTTP clients.

Syntax

`true` or `false`. The default is `false`.

Usage

Directory browsing allows a client to see a list of files in a directory when the specified URL ends in a directory name and the directory does not include a welcome file. If directory browsing is not enabled for the directory, EAServer returns HTTP 500 errors for these requests.

To enable browsing, you must set this property to true and also list the directory trees that can be browsed by setting the properties com.sybase.jaguar.server.http.dirbrowseinclude and com.sybase.jaguar.server.http.dirbrowsewebappinclude.

Restart the server for the changes to take effect.

See also com.sybase.jaguar.server.http.dirbrowseinclude, com.sybase.jaguar.server.http.dirbrowsewebappinclude

com.sybase.jaguar.server.http.dirbrowseinclude

Description This property is no longer supported. If directory browsing is enabled, specify the directories under the document root that can be browsed using the com.sybase.jaguar.server.http.dirbrowsewebappinclude property.

Syntax The value is ignored.

com.sybase.jaguar.server.http.dirbrowsewebappinclude

Description If directory browsing is enabled, specifies Web application subdirectories that can be browsed.

Syntax A list of Web applications and browseable directories within each web application. Each entry in the list must be in the form:

(*WebApp*, *dir-list*)

Where:

- *WebApp* is the Web application name. Use `*` to specify all Web applications; in this case, the specified directories can be browsed in any Web application where they exist.
- *dir-list* is a comma-separated list of directories which must begin with a `/`. The contents of each specified directory and its subdirectories can be browsed. To allow browsing of all directories, enter `/*`.

For example, to allow browsing of *images* in the Retail Web application and *docs* and *images* in the Wholesale Web application, specify:

```
(Retail, /images), (Wholesale, /docs, /images)
```

To allow browsing of *images* and *help* directories in any Web application, specify:

```
(*, /images, /help)
```

To allow browsing of any directory in any Web application, specify:

```
(*, /*)
```

Usage	Browsing must be enabled by setting the com.sybase.jaguar.server.http.dirbrowseenable property. Restart the server for the changes to take effect after changing these properties. The special Web application directories WEB-INF and META-INF cannot be browsed even if specified because the Java Servlet specification does not allow servlet containers to return content from these directories.
See also	com.sybase.jaguar.server.http.dirbrowseenable, com.sybase.jaguar.webapplication.welcome-file-list

com.sybase.jaguar.server.http.disablechunkedtransfer

Description	Specifies whether to send HTTP 1.1 client responses as chunked streams or in byte-serving mode. If not set, responses are sent as chunked streams.
Syntax	`true` or `false`. The default is `false`.

com.sybase.jaguar.server.http.docroot

Description	Specifies the HTTP document root directory.
Syntax	The path to the directory. You can use *%JAGUAR%* or *$JAGUAR* to substitute for the EAServer installation directory. The default is the EAServer *html* subdirectory.

com.sybase.jaguar.server.http.domainname

Description	Specifies the host and/or domain name for the proxy HTTP server.
Syntax	The host or domain name; for example, `sybooks.sybase.com` or `sybase.com`. There is no default.
Usage	Set this only if you are configuring the redirection URL for use with a Web proxy, as described in “Configuring redirection addresses when using a proxy server” on page 27.
See also	com.sybase.jaguar.server.http.httpport, com.sybase.jaguar.server.http.httpsport, com.sybase.jaguar.server.http.proxyprotocol, com.sybase.jaguar.webapplication.httpdomain.override

com.sybase.jaguar.server.http.elffenable

Description — Specifies whether the extended log file format is used to write information to the request log.

Syntax — `true` or `false`. The default is `false`.

See also — com.sybase.jaguar.server.http.elffitems

com.sybase.jaguar.server.http.elffitems

Description — If the HTTP request log uses extended log file format (ELFF), specifies the items and the order in which to write them to the request log.

Syntax — A comma-separated list of items. Table B-20 lists the felds that can be included in the list and the headers that identify each field in the log:

Table B-20: Extended log file format items

Field name	Log header
hostaddr	s-ip
clientaddr	c-ip
date	date
time	time
reqline	cs-request
status	cs-status
length	cs-bytes
referer	cs(Referer)
cookie	cs(Cookie)
user-agent	cs(User-Agent)
method	cs-method
uri-stem	cs-uri-stem
http_version	c-httpversion
host	s-host
post	s-port

Usage — This property is used only when you set com.sybase.jaguar.server.http.elffenable to `true`.

The default is:

clientaddr, date, time, reqline, status, length, cookie, referer

Which generates something like this in the log file:

```
#Version: 1.0
#Date: 2001-12-31 04:10:00
#Fields: c-ip date time cs-request cs-status cs-bytes cs(Cookie) cs(Referer)
120.0.0.1 2001-12-31 04:10:01 "GET /index.html HTTP/1.0" 304 0 -
"http://localhost:8080/index.html"
120.0.0.1 2001-12-31 04:10:02 "GET /images/tiny.jpg HTTP/1.0" 304 0 -
"http://localhost:8080/index.html"
```

See also com.sybase.jaguar.server.http.elffenable

com.sybase.jaguar.server.http.errorlogname

Description Specifies the HTTP error log file name.

Syntax The path and file name. The path can be a full path, or relative to the EAServer *bin* directory. The default is:

server`httperror.log`

Where server is the *server* name.

See also com.sybase.jaguar.server.http.errorlogsize, com.sybase.jaguar.server.http.errorlogtruncate

com.sybase.jaguar.server.http.errorlogsize

Description Specifies the maximum size of the HTTP error log.

Syntax The size in bytes. No value means there is no size limit. If the maximum size is reached, the existing log is closed and renamed and a new log is created.

See also com.sybase.jaguar.server.http.errorlogname, com.sybase.jaguar.server.http.errorlogtruncate

com.sybase.jaguar.server.http.errorlogtruncate

Description Specifies whether the HTTP error log is truncated when the server is restarted.

Syntax `true` or `false`. The default is `false`.

See also com.sybase.jaguar.server.http.errorlogname, com.sybase.jaguar.server.http.errorlogsize

com.sybase.jaguar.server.http.force.close

Description	Specifies whether HTTP-tunnelled IIOP connections should be closed after sending each IIOP response.
Syntax	`true` or `false`. The default is `false`.
Usage	When debugging proxy configurations, you can set this property to `true` if it helps your debugging efforts. This setting degrades server performance, and we suggest that you use it only for debugging purposes.

com.sybase.jaguar.server.http.httpport

Description	When a domain name is specified by setting com.sybase.jaguar.server.http.domainname, the port for HTTP redirection URLs.
Syntax	The port number. If not specified, the default is 80.
Usage	Set this only if you are configuring the redirection URL for use with a Web proxy, as described in “Configuring redirection addresses when using a proxy server” on page 27.
See also	com.sybase.jaguar.server.http.domainname, com.sybase.jaguar.server.http.httpsport, com.sybase.jaguar.server.http.proxyprotocol

com.sybase.jaguar.server.http.httpsport

Description	When a domain name is specified by setting com.sybase.jaguar.server.http.domainname, the port for HTTP redirection URLs.
Syntax	The port number. If not specified, the default is 80.
Usage	Set this only if you are configuring the redirection URL for use with a Web proxy, as described in “Configuring redirection addresses when using a proxy server” on page 27.
See also	com.sybase.jaguar.server.http.domainname, com.sybase.jaguar.server.http.httpport, com.sybase.jaguar.server.http.proxyprotocol

com.sybase.jaguar.server.http.maxthreads

Description	Specifies the maximum number of threads to handle HTTP client requests.
Syntax	A positive integer. The default is 25.
See also	com.sybase.jaguar.server.maxthreads

com.sybase.jaguar.server.http.proxyprotocol

Description	When a domain name is specified by setting com.sybase.jaguar.server.http.domainname, the protocol for redirection URLs.
Syntax	The protocol; for example, HTTP or HTTPS. The default is the protocol of the original request.
Usage	Set this only if you are configuring the redirection URL for use with a Web proxy, as described in "Configuring redirection addresses when using a proxy server" on page 27.
See also	com.sybase.jaguar.server.http.domainname, com.sybase.jaguar.server.http.httpport, com.sybase.jaguar.server.http.httpsport

com.sybase.jaguar.server.http.requestlogenable

Description	Enables and disables HTTP request logging.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.server.http.requestlogname, com.sybase.jaguar.server.http.requestlogsize, com.sybase.jaguar.server.http.requestlogtruncate

com.sybase.jaguar.server.http.requestlogname

Description	Specifies the HTTP request log file name.
Syntax	The path and file name. The path can be a full path, or relative to the EAServer *bin* directory. The default is: `*server*httprequest.log` Where server is the *server* name.

See also com.sybase.jaguar.server.http.requestlogenable,
com.sybase.jaguar.server.http.requestlogsize,
com.sybase.jaguar.server.http.requestlogtruncate

com.sybase.jaguar.server.http.requestlogsize

Description Specifies the maximum size of the HTTP request log.

Syntax The size in bytes. No value means there is no size limit. If the maximum size is reached, the existing log is closed and renamed and a new log is created.

See also com.sybase.jaguar.server.http.requestlogenable,
com.sybase.jaguar.server.http.requestlogname,
com.sybase.jaguar.server.http.requestlogtruncate

com.sybase.jaguar.server.http.requestlogtruncate

Description Specifies whether the HTTP request log is truncated when the server is restarted.

Syntax `true` or `false`. The default is `false`.

See also com.sybase.jaguar.server.http.requestlogname,
com.sybase.jaguar.server.http.requestlogsize

com.sybase.jaguar.server.http.sendserverheader

Description Specifies whether to EAServer should add the "Server" response header field to each HTTP response. This optional HTTP response header field contains a description of the server software.

Syntax `true` or `false`. The default of `false` specifies omission of the "Server" field in the response header.

com.sybase.jaguar.server.http.servletlogenable

Description Obsolete since version 5.0. Configure the log profile properties to discard messages logged from the servlet engine.

Syntax N/A.

See also	Log profile properties on page 460

com.sybase.jaguar.server.http.servletlogname

Description	Obsolete since version 5.0. Use the log profile properties to specify a file name.
Syntax	N/A.
See also	Log profile properties on page 460

com.sybase.jaguar.server.http.servletlogsize

Description	Obsolete since version 5.0. Use the log profile properties to specify a maximum file size.
Syntax	N/A.
See also	Log profile properties on page 460

com.sybase.jaguar.server.http.servletlogtruncate

Description	Specifies whether the HTTP servlet log is truncated when the server is restarted.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.server.http.servletlogname, com.sybase.jaguar.server.http.servletlogsize

com.sybase.jaguar.server.http.sso

Description	Specifies whether sign-on occurs externally. Set this property to true if you are using an external single-sign on provider.
Syntax	`true` or `false`. The default is `false`.
See also	“Supporting external single sign-on providers” in the *EAServer Security Administration and Programming Guide*.

com.sybase.jaguar.server.http.statlogenable

Description: Enables and disables HTTP statistics logging.

Syntax: `true` or `false`. The default is `false`.

See also: com.sybase.jaguar.server.http.statlogfrequency, com.sybase.jaguar.server.http.statlogname

com.sybase.jaguar.server.http.statlogfrequency

Description: When HTTP statistics logging is enabled, specifies the update frequency.

Syntax: An integer specifying the interval between updates, in seconds. The default is 36000.

See also: com.sybase.jaguar.server.http.statlogenable, com.sybase.jaguar.server.http.statlogname

com.sybase.jaguar.server.http.statlogname

Description: Specifies the HTTP statistics log file name.

Syntax: The path and file name. The path can be a full path, or relative to the EAServer *bin* directory. The default is:

```
server httpstat.dat
```

Where server is the *server* name.

See also: com.sybase.jaguar.server.http.statlogenable, com.sybase.jaguar.server.http.statlogfrequency

com.sybase.jaguar.server.httpservices

Description: When PowerDynamo execution is enabled, specifies the evaluation order of PowerDynamo and servlet path mappings.

Syntax

Value	To indicate
`dynamo,servlet` (the default)	PowerDynamo request path mappings take precedence over servlet path mappings.
`servlet,dynamo`	The reverse of the above.

See also	com.sybase.jaguar.server.dynamo.exec

com.sybase.jaguar.server.iiop.log

Description	Enables and disables IIOP logging in the server.
Syntax	`true` or `false`. The default is `false`.
Usage	IIOP logging can be useful for debugging, but generates a tremendous amount of log output.

com.sybase.jaguar.server.iiop.log.ac

Description	Enables and disables IIOP logging for events that occur after the server begins accepting messages.
Syntax	`true` or `false`. The default is `false`.
Usage	This generates less log output than com.sybase.jaguar.server.iiop.log because it does not log messages during server start-up.

com.sybase.jaguar.server.intffilename

Description	Specifies the interfaces file name used by Open Client Client-Library connection caches.
Syntax	The file name. For UNIX platforms, the default is *$JAGUAR/interfaces*. For Windows, the default is *%JAGUAR%\ini\sql.ini*.
Usage	This interfaces file is used only by the Client-Library implementation provided with EAServer for use in components that use Client-Library connection caches.

com.sybase.jaguar.server.jaas.config

Description	Specifies the JAAS configuration file.
Syntax	The path to the JAAS configuration file.
Usage	For more information on JAAS, see Chapter 11, “Using the JAAS API,” in the *EAServer Security Administration and Programming Guide*.

See also com.sybase.jaguar.server.jaas.section

com.sybase.jaguar.server.jaas.section

Description Specifies the section name in the JAAS configuration file to be used for this server.

Syntax The section name. If not specified, the default is the name of the server.

Usage Setting this property allows you to use the same JAAS configuration setting in multiple servers. For more information on JAAS, see Chapter 11, "Using the JAAS API," in the *EAServer Security Administration and Programming Guide*.

See also com.sybase.jaguar.server.jaas.config

com.sybase.jaguar.server.jagadminpassword

Description Specifies the password for the built-in jagadmin administrative account.

Syntax The password text. Values are encrypted in the repository.

com.sybase.jaguar.server.jagmgr.DOMFactoryChoice

Description Specifies the DOM parser configuration displayed in EAServer Manager.

Syntax

Value	To indicate
0	No parser
1	Platform default
2	The parser specified by the com.sybase.jaguar.server.DOMfactory property

See also com.sybase.jaguar.server.DOMfactory, com.sybase.jaguar.server.jagmgr.SAXFactoryChoice, com.sybase.jaguar.server.jagmgr.XSLTFactoryChoice

com.sybase.jaguar.server.jagmgr.SAXFactoryChoice

Description — Specifies the SAX parser configuration displayed in EAServer Manager.

Syntax

Value	To indicate
0	No parser
1	Platform default
2	The parser specified by the com.sybase.jaguar.server.SAXfactory property

See also — com.sybase.jaguar.server.SAXfactory, com.sybase.jaguar.server.jagmgr.DOMFactoryChoice, com.sybase.jaguar.server.jagmgr.XSLTFactoryChoice

com.sybase.jaguar.server.jagmgr.XSLTFactoryChoice

Description — Specifies the XSLT parser configuration displayed in EAServer Manager.

Syntax

Value	To indicate
0	No parser
1	Platform default
2	The parser specified by the com.sybase.jaguar.server.XSLTfactory property

See also — com.sybase.jaguar.server.XSLTfactory, com.sybase.jaguar.server.jagmgr.DOMFactoryChoice, com.sybase.jaguar.server.jagmgr.SAXFactoryChoice

com.sybase.jaguar.server.java.classes

Description — Specifies Java classes and JAR files to be loaded by the server's custom class loader.

Syntax — Same as for com.sybase.jaguar.application.java.classes.

Usage — See "Custom class lists for packages, applications, or servers" in Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*.

In EAServer Manager, set this property using the Java Classes tab in the Package Properties dialog box.

See also com.sybase.jaguar.server.classloader.debug, com.sybase.jaguar.component.java.classes, com.sybase.jaguar.application.java.classes, com.sybase.jaguar.package.java.classes, com.sybase.jaguar.webapplication.java.classes

com.sybase.jaguar.server.jcm.trace

Description Enables tracing of Java Connection Manager (JCM) activity.

Syntax The get connection call state, which can be one of:

State	Description
true	Tracing enabled. Trace information is written to the server log. Restart the server for the change to take effect.
false	The default. Tracing is disabled.

com.sybase.jaguar.server.jpda.port

Description Specifies the port number for JDPA remote debugging connections.

Syntax The port number.

See also com.sybase.jaguar.server.jvm.debug.options

com.sybase.jaguar.server.jpda.suspend

Description When using the JPDA remote debugging interface, allows you to pause the server at startup so that breakpoints can be set in Java code that executes at startup time.

Syntax `true` or `false`. In the default configuration, the value is `${JPDASUSPEND}`, which evaluates to the JPDASUSPEND environment variable as set in the server startup scripts. The default configuration allows you to set this property with the -jpdasuspend command line option. See “Starting the server” on page 49 for more information.

Usage After the server pauses, you must resume execution with your remote debugger.

com.sybase.jaguar.server.jta.tranTableSize

Description — Specifies the size of the in-memory table that the transaction manager uses to cache information about pending transactions.

Syntax — An integer value. The default is 1024. For best performance, set the value to at least one-half of the maximum number of simultaneous transactions expected in your application.

com.sybase.jaguar.server.jvm.bootclasspath

Description — Specifies the directories and JAR files in the boot classpath search list for the Java virtual machine. Classes loaded from these locations can override the core Java runtime classes. In most cases, the value should match the com.sybase.jaguar.server.jvm.classpath setting.

Syntax — The default is `${BOOTCLASSPATH}`, which is replaced by the valur of the BOOTCLASSPATH environment variable at runtime. If you add entries or use a different value, the resultant list must consist of directories and the full paths to JAR files. On Windows platforms, use a semicolon (;) to separate entries. On UNIX platforms, use a colon (:). For example, this setting uses the default prefixed with a different JAR file:

```
d:\devstuff\experimental.jar;${BOOTCLASSPATH}
```

See also — com.sybase.jaguar.server.jvm.bootclasspath.jars, com.sybase.jaguar.server.jvm.classpath

com.sybase.jaguar.server.jvm.bootclasspath.jars

Description — Specifies additional JAR files to be included in the BOOTCLASSPATH setting for the server's Java virtual machine.

Syntax — A comma separated list of JAR file names, which must be located in the EAServer *java/classes* subdirectory, for example:

```
jaxrpc-api.jar,commons-logging.jar,log4j.jar
```

Usage — Setting this property avoids the limit on environment variable size. If the BOOTCLASSPATH setting exceeds the size limit, the Windows serverstart batch file may fail. Although this problem occurs on Windows, the setting can be used on all platforms.

JAR files located in the *java/lib* subdirectory are automatically added to the CLASSPATH and BOOTCLASSPATH environment variable when the server starts.

See also: com.sybase.jaguar.server.jvm.bootclasspath, com.sybase.jaguar.server.jvm.classpath.jars

com.sybase.jaguar.server.jvm.bootlibpath

Description: Specifies the directory search path to load native libraries used by Java classes.

Syntax: The syntax is the same as for the platform PATH environment variable setting. The default is `${BOOTLIBRARYPATH}`, which is replaced by the value of the BOOTLIBRARYPATH variable at runtime.

See also: com.sybase.jaguar.server.jvm.bootclasspath, com.sybase.jaguar.server.jvm.classpath

com.sybase.jaguar.server.jvm.classloader

Description: Specifies which version of the EAServer custom class loader to use.

Syntax: A value of `JaguarClassLoaderv2` specifies version 2.0 of the EAServer custom class loader. Any other value specifies version 1.0.

Usage: EAServer uses customized Java class loaders to allow *hot refresh* of Web application classes and Java components without restarting the server. EAServer versions earlier than 5.1 use version 1.0 of the custom class loader. Beginning in EAServer 5.1, you can use the version 2.0 class loader by setting this property. The newer class loader provides additional configuration options and improved diagnostics. For backward compatibility with applications developed with EAServer 5.1, the default EAServer configuration uses the version 1.0 class loader.

See also: Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.server.jvm.classpath

Description — Specifies the classpath for the Java virtual machine, which is the list of directories and JAR files that are searched to load classes. These locations are searched after the locations specified in com.sybase.jaguar.server.jvm.bootclasspath and cannot override the core Java runtime classes.

Syntax — The default is `${CLASSPATH}`, which is replaced by the valur of the CLASSPATH environment variable at runtime. If you add entries or use a different value, the resultant list must consist of directories and the full paths to JAR files. On Windows platforms, use a semicolon (;) to separate entries. On UNIX platforms, use a colon (:). For example, this setting uses the default prefixed with a different JAR file:

```
d:\devstuff\experimental.jar;${BOOTCLASSPATH}
```

See also — com.sybase.jaguar.server.jvm.classpath.jars, com.sybase.jaguar.server.jvm.bootclasspath

com.sybase.jaguar.server.jvm.classpath.jars

Description — Specifies additional JAR files to be included in the CLASSPATH setting for the server's Java virtual machine.

Syntax — A comma separated list of JAR file names, which must be located in the EAServer *java/classes* subdirectory, for example:

```
jaxrpc-api.jar,commons-logging.jar,log4j.jar
```

Usage — Setting this property avoids the limit on environment variable size. If the CLASSPATH setting exceeds the size limit, the Windows serverstart batch file may fail. Although this problem occurs on Windows, the setting can be used on all platforms.

JAR files located in the *java/lib* subdirectory are automatically added to the CLASSPATH and BOOTCLASSPATH environment variable when the server starts.

See also — com.sybase.jaguar.server.jvm.bootclasspath

com.sybase.jaguar.server.jvm.debug.options

Description — Specifies additional JVM options to configure in-server Java debugging.

Syntax — The options string, as it would be passed on the command line to the java executable. When you start the server in debug mode, these options are passed in addition to those specified by the com.sybase.jaguar.server.jvm.options property.

See also — com.sybase.jaguar.server.jpda.port, com.sybase.jaguar.server.jvm.options

com.sybase.jaguar.server.jvm.displayOptions

Description — When set to true, the server logs the JVM options when starting.

Syntax — `true` or `false`. The default is `false`.

See also — com.sybase.jaguar.server.jvm.options, com.sybase.jaguar.server.jvm.debug.options, com.sybase.jaguar.server.jvm.maxHeapSize, com.sybase.jaguar.server.jvm.minHeapSize

com.sybase.jaguar.server.jvm.maxHeapSize

Description — Specifies the maximum heap size for the Java virtual machine.

Syntax — The size in bytes, kilobytes, or megabytes, as described in the following table:

Heap size value syntax	To indicate
n`M` or *n*`m`	*n* megabytes, for example: `512M`
n`K` or *n*`k`	*n* kilobytes, for example: `1024K`
n	*n* bytes, for example: `536870912`

For JDK 1.3, the value must be a multiple of 1024 greater than 2 megabytes. The default depends on the JDK version:

JDK version	Max heap size default
1.3 on Windows	500MB
1.3 on Solaris	1024MB
1.3 on HP-UX	1024MB

JDK version	Max heap size default
1.3 on AIX	1500MB
1.3 on Linux (using the IBM JDK 1.3)	1500MB

Usage — This property corresponds to the `-Xmx` option for the Java interpreter. If you specify a `-Xmx` setting in the value of the property com.sybase.jaguar.server.jvm.options, it takes precedence over the value of this property.

See also — com.sybase.jaguar.server.jvm.options, com.sybase.jaguar.server.jvm.minHeapSize

com.sybase.jaguar.server.jvm.minHeapSize

Description — Specifies the minimum heap size for the Java virtual machine.

Syntax — Same as for com.sybase.jaguar.server.jvm.maxHeapSize.

Usage — This property corresponds to the `-Xms` option for the Java interpreter. If you specify a `-Xms` setting in the value of the property com.sybase.jaguar.server.jvm.options, it takes precedence over the value of this property.

See also — com.sybase.jaguar.server.jvm.options, com.sybase.jaguar.server.jvm.maxHeapSize

com.sybase.jaguar.server.jvm.nojit

Description — When using the Client Hotspot Java virtual machine (VM), specifies whether the just-in-time (JIT) Java compiler is disabled.

Syntax — `true` or `false`. In EAServer 4.1 and later, the default for new servers is `false`. In releases prior to 4.1, the default is `true`.

Usage — This property has no effect if the server is started with the Server Hotspot VM. The server VM is determined by the command-line options used to start the server or to install it as a Windows service.

See also — “Starting the server” on page 49

com.sybase.jaguar.server.jvm.options

Description — Specifies options to be passed to the server's Java virtual machine.

Syntax — The options string, in a format similar to command-line options. To specify multiple options, separate the values with commas, for example:

```
com.sybase.jaguar.server.jvm.options=-DMyVariable=true,-DMyOtherVariable=no
```

To specify an option setting that includes commas, wrap the setting in single quotes, for example:

```
com.sybase.jaguar.server.jvm.options='comment=value1, with a comment'
```

The default is:

```
-Djava.protocol.handler.pkgs=com.sybase.jaguar.net
```

The default configures the EAServer HTTPS connection protocol handler as the default for HTTPS protocol javax.net.URLConnection instances.

Usage — This property allows you to configure the Java Virtual Machine options directly. See the JDK documentation for information on setting options.

You can set properties specific to a particular JDK version by setting the properties com.sybase.jaguar.server.jvm13.options or com.sybase.jaguar.server.jvm14.options. These properties allow you to use the same server properties when the server requires different JVM options depending on which JDK version is specified at start-up.

Some other properties, referenced in the See also section, configure specific JVM options; settings from these other properties are added to the JVM options when EAServer creates the JVM. For troubleshooting these settings, you can set the com.sybase.jaguar.server.jvm.displayOptions property so the server logs the complete options string at start-up.

See also — com.sybase.jaguar.server.jvm.debug.options, com.sybase.jaguar.server.jvm.displayOptions, com.sybase.jaguar.server.jvm.maxHeapSize, com.sybase.jaguar.server.jvm.minHeapSize, com.sybase.jaguar.server.jvm13.options, com.sybase.jaguar.server.jvm14.options

com.sybase.jaguar.server.jvm13.options

Description — Specifies additional Java VM options when the server is started with JDK 1.3.

Syntax — The syntax is the same as com.sybase.jaguar.server.jvm.options. The JDK 1.3 options are appended to those set by the com.sybase.jaguar.server.jvm.options property.

com.sybase.jaguar.server.jvm14.options

Description — Specifies additional Java VM options when the server is started with JDK 1.4.

Syntax — The syntax is the same as com.sybase.jaguar.server.jvm.options. The JDK 1.4 options are appended to those set by the com.sybase.jaguar.server.jvm.options property.

com.sybase.jaguar.server.jvm.verbose

Description — Enables and disables verbose logging of Java system class loader activity.

Syntax — `true` or `false`. The default is `false`.

com.sybase.jaguar.server.jvm.verboseGC

Description — Enables and disables verbose logging of Java garbage collection.

Syntax — `true` or `false`. The default is `false`.

com.sybase.jaguar.server.jvm.verboseJNI

Description — Enables and disables verbose logging of Java Native Interface (JNI) method linking and execution.

Syntax — `true` or `false`. The default is `false`.

com.sybase.jaguar.server.keytabfile

Description — Specifies the server's key tab file, which is used for interserver authentication in a cluster.

Syntax — The file name. The file is generated when you create a cluster, and there is normally no need to set this property manually.

com.sybase.jaguar.server.language

Description	Specifies the language (locale) in which messages are logged.
Syntax	The language or locale name. The default is `us_english`.

com.sybase.jaguar.server.libdir

Description	Specifies the location from which the server loads DLLs or shared libraries for C++ and C components.
Syntax	The directory name. The default is the EAServer *cpplib* subdirectory.

com.sybase.jaguar.server.listeners

Description	Specifies the server's network listeners as a comma-separated list.
Syntax	*listener1*, *listener2*, ... Where *listener1*, *listener2*, and so forth are listener names defined in the repository.
See also	Listener properties on page 456

com.sybase.jaguar.server.logfilename

Description	Obsolete since version 5.0. Use the log profile properties to specify a file name.
Syntax	N/A.
See also	Log profile properties on page 460

com.sybase.jaguar.server.logfilesize

Description	Obsolete since version 5.0. Use the log profile properties to specify the maximum file size.
Syntax	N/A.
See also	Log profile properties on page 460

com.sybase.jaguar.server.logging.profile.debug

Description	The log profile used when the server is started in debug mode.
Syntax	The log profile name.
See also	Log profile properties on page 460, com.sybase.jaguar.server.logging.profile.prod

com.sybase.jaguar.server.logging.profile.prod

Description	The log profile used when the server is started in production mode.
Syntax	The log profile name.
See also	Log profile properties on page 460, com.sybase.jaguar.server.logging.profile.debug

com.sybase.jaguar.server.logspid

Description	Specifies whether the thread identifier is included with log messages.
Syntax	`true` or `false`. When set to `true`, the thread identifier is included in each log message record. The default is `false`.
Usage	To record the thread identifier in the log, you must also configure a log record formatter that includes a placeholder for the thread identifier (`%TI`) in the output pattern. See "Formatter properties" on page 66 for more information.
See also	"Configuring log profiles" on page 56

com.sybase.jaguar.server.lwc

Description	Enables the EAServer lightweight container (LWC) for intercomponent EJB invocations or calls to EJBs from servlets and JSPs hosted in the same server. For more information, see "Lightweight container" in the *EAServer Performance and Tuning Guide*.
Syntax	`true` or `false`. A value of `true` allows you to enable LWC for indivicual components by setting the component property com.sybase.jaguar.component.lwc. The default of `false` disables LWC for all components regardless of the component setting.

See also com.sybase.jaguar.server.lwc.enableSkeletons, com.sybase.jaguar.component.lwc

com.sybase.jaguar.server.lwc.debug

Description Enables logging of debug (tracing) output from the lightweight container.

Syntax `true` or `false`. The default of `false` disables debug output.

See also com.sybase.jaguar.server.lwc

com.sybase.jaguar.server.lwc.enableSkeletons

Description If com.sybase.jaguar.server.lwc is true, enables LWC calls to EJB components from servlets and JSPs hosted in the same server. Such calls are not supported unless this option is set.

Syntax `true` or `false`. A value of `true` allows LWC calls to components from servlets and JSPs. The default is `false`.

See also com.sybase.jaguar.server.lwc, com.sybase.jaguar.component.lwc.enableSkeletons

com.sybase.jaguar.server.masp.zero-success

Description Configures the return status for Methods As Stored Procedures (MASP) invocations.

Syntax

Value	To indicate
`false` (the default)	A return value of 1 indicates success; a value of 0 indicates failure.
`true`	Reverses the meanings of 0 and 1 return values.

See also Appendix A, "Executing Methods As Stored Procedures," in the *EAServer Programmer's Guide*

com.sybase.jaguar.server.maxconnections

Description	Specifies the maximum number of simultaneous IIOP connections.
Syntax	A positive integer. The default is 30.

com.sybase.jaguar.server.maxthreads

Description	Specifies the maximum number of simultaneous threads.
Syntax	A positive integer. The default is 50.

com.sybase.jaguar.server.memory.alarmLimit

Description	The percentage of system memory that can be used before the server begins blocking external requests.
Syntax	The percentage as a decimal number. If not set, the default is 70.
See also	com.sybase.jaguar.server.memory.lockLimit
	Chapter 9, “Using the Performance Monitor,” in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.server.memory.lockLimit

Description	The percentage of system memory that can be used before the server begins blocking external requests and begins cancelling in-process requests to bring usage below the specified critical threshold.
Syntax	The percentage as a decimal number. If not set, the default is 90.
See also	com.sybase.jaguar.server.memory.alarmLimit
	Chapter 9, “Using the Performance Monitor,” in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.server.name

Description	Specifies the server name.
Syntax	The name text.

com.sybase.jaguar.server.nameservice

Description	Specifies the default NameServiceURL parameter for the C++ and Java ORB when run in the server. Also, when the com.sybase.jaguar.server.CosNaming.nameserver property is `false`, specifies the server to connect to for naming services.
Syntax	The IIOP URL for naming services. The default is `iiop://0:0`. The default is appropriate if the server provides naming services to itself.
See also	com.sybase.jaguar.server.CosNaming.nameserver

com.sybase.jaguar.server.nativemutex

Description	Specifies whether native (operating system) thread mutexes are used in the server.
Syntax	`true` or `false`. The default is `true`.

com.sybase.jaguar.server.netbufsize

Description	Specifies the default size of the network buffer.
Syntax	The size in bytes. The default is 2048.

com.sybase.jaguar.server.packages

Description	Specifies the packages that are installed in this server.
Syntax	A comma-separated list of package names.
Usage	Packages may be installed in a server directly using this property or as part of an application using the com.sybase.jaguar.server.applications property.
See also	com.sybase.jaguar.server.applications, Package properties on page 483

com.sybase.jaguar.server.perfmonitor.logfilename

Description	Specifies the name and path to the EAServer performance monitor log file.

Syntax

The path and file name. The path can be a full path, or relative to the EAServer *bin* directory. The default is:

```
server_performance.log
```

Where server is the *server* name.

com.sybase.jaguar.server.roles

Description

Specifies server-wide access control restrictions.

Syntax

The name of a valid EAServer role.

Usage

If you set this property to a name of a role, only users that are members of that role can access the methods, components, and packages hosted on that server. For example, if you set this property to "MyRole", only users that are members of "MyRole" can access the components hosted on the server.

This property does not affect Web applications, or the resources it contains.

com.sybase.jaguar.server.roleservice

Description

Specifies the name of a custom component that evaluates a user's role membership to control access to components and HTTP URLs.

Syntax

package/component

Where *package/component* is the name of an EAServer package/component.

com.sybase.jaguar.server.SAXfactory

Description

Specifies the class name for a custom SAX XML parser factory class.

Syntax

The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.server.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.

See also

com.sybase.jaguar.server.jagmgr.SAXFactoryChoice, com.sybase.jaguar.server.DOMfactory, com.sybase.jaguar.server.XSLTfactory

com.sybase.jaguar.server.security.identities

Description — Specifies the list of trusted identities used for incoming component invocations when propagating client credentials from another server.

Syntax — *t1*,*t2*,*t3*,...

Where *t1*, *t2*, and so forth are identity names that are defined in the repository.

See also — com.sybase.jaguar.server.security.identity, Security properties

com.sybase.jaguar.server.security.identity

Description — Specifies the identity used for outgoing component invocations when propagating client credentials to another server.

Syntax — The name of an identity that is defined in the repository.

See also — com.sybase.jaguar.server.security.identities, Security properties

com.sybase.jaguar.server.services

Description — A list of components that run as service components in the server.

Syntax — A comma-separated list of components; for example:

```
CtsComponents/MessageService
```

By default, one thread runs per service. To specify multiple threads for a service, enter the number of threads in brackets after the component name. For example:

```
YourPackage/YourService[10]
```

See also — Chapter 33, "Creating Service Components," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.server.servlet.aliases

Description — Specifies the aliases (prefixes) for execution of servlets that are installed directly in this server.

Syntax — A comma-separated list of aliases. If not set, the default alias is "servlet".

Usage

Servlets that are installed directly in the server using the com.sybase.jaguar.server.servlets property are executed using request paths of the form:

```
/alias/servlet-name
```

Or:

```
/alias/servlet-class
```

Where *alias* is the servlet alias, *servlet-name* is the servlet name, and servlet-class is the servlet class name in Java dot notation. Class-name requests can be disabled with the com.sybase.jaguar.server.servlet.class-name-req property.

This property does not affect servlets or JSPs that are installed in a Web application.

See also

com.sybase.jaguar.server.servlets, com.sybase.jaguar.server.servlet.class-name-req, com.sybase.jaguar.server.webapplications

com.sybase.jaguar.server.servlet.class-name-req

Description

Specifies whether servlets that are installed directly in this server can be executed by specifying the class name in the URL.

Syntax

`true` or `false`. The default is `true`.

Usage

This property affects only servlets that are installed directly in the server (those specified by the com.sybase.jaguar.server.servlets property), not servlets that are installed in a Web application.

See also

com.sybase.jaguar.server.servlet.aliases, com.sybase.jaguar.server.servlets

com.sybase.jaguar.server.servlet.context-param

Description

Specifies the context initialization parameters for the servlets that are installed directly in this server.

Syntax

A string of the form:

entry1, *entry2*, ...

Where *entry1*, *entry2*, and so forth are of the form:

(description=*desc*,value=*value*,name=*name*)

Where:

- *desc* is an optional comment describing how the parameter is to be set.
- *value* is the value of the parameter.
- *name* is the parameter's name.

Usage — This property allows you to specify context initialization parameters for servlets that are installed directly in the server; that is, those specified by the com.sybase.jaguar.server.servlets property.

See also — com.sybase.jaguar.server.servlets, com.sybase.jaguar.webapplication.context-param

com.sybase.jaguar.server.servlet.destroy-wait-time

Description — The number of seconds that EAServer should wait for servlet service calls to return before calling the servlet destroy method.

Syntax — The timeout value in seconds. A value of 0 indicates infinity.

Usage — This property affects all servlets installed directly in the server; that is, those specified by the com.sybase.jaguar.server.servlets property. It does not affect servlets installed in a Web application.

You can override the server-wide setting for individual servlets by setting the com.sybase.jaguar.servlet.destroy.wait-time property.

See also — com.sybase.jaguar.server.servlets, com.sybase.jaguar.server.servlet.init-timeout

com.sybase.jaguar.server.servlet.error-page

Description — Identifies server-level error pages, which allow you to customize the response that the server sends to clients when an error occurs. You can specify HTML files to send in response to HTTP error codes and to Java exceptions thrown in JSPs or servlets.

Syntax — A comma-delimited list of complex properties. Each property includes a location, and either an error code or an exception field.

(location=*/file.jsp*,error-code=(*code*),
(location=*/exception.htm*,exception=*java.lang.Exception*)

Where:

- The locations of *file.jsp* and *exception.htm* are relative to the HTML document root.
- *code* is the error code that triggers the error page.
- *java.lang.Exception* is the fully-qualified Java class name of the exception that triggers the error page.

See also "Welcome and error page specifications" in Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.server.servlet.exec

Description Enables and disables servlet execution.

Syntax `true` or `false`. The default is `true`, which enables servlet execution.

com.sybase.jaguar.server.servlet.init-timeout

Description Specifies how long the server should wait for each servlet's init method to return.

Syntax Same as com.sybase.jaguar.servlet.init.timeout.

Usage This property affects servlets installed directly in the server; that is, those specified by the com.sybase.jaguar.server.servlets property. For servlets installed in a Web application, set the Web application property com.sybase.jaguar.webapplication.init-timeout.

You can override the value for individual servlets by setting the servlet property com.sybase.jaguar.servlet.init.timeout.

See also com.sybase.jaguar.server.servlet.destroy-wait-time, com.sybase.jaguar.servlet.init.timeout, com.sybase.jaguar.webapplication.init-timeout

com.sybase.jaguar.server.servlet.max4kbuffers

Description Specifies the number of internal 4K servlet response buffers.

Syntax A positive integer. The default is 128. You can override the default by setting the value to a positive integer. A value of 0 means buffers are never pooled.

Usage

Internally, EAServer uses 4K and 8K temporary buffers when assembling servlet responses. These buffers are pooled and reused to avoid the overhead of repeated buffer allocation and garbage collection.

The required buffers are allocated on an as-needed basis, rather than being preallocated as server startup. Once allocated, buffers are pooled and reused until the specified size is reached. If a peak in client activity requires more buffers than the pool size, additional buffers are allocated, then released for garbage collection after use.

The default configuration suffices for most applications. If the buffer pool size is too small, performance.may decline due to allocation of new buffers. Allocation is costly because the Java VM initializes the allocated byte arrays to 0, which is not required by EAServer. Garbage collection is also costly. On the other hand, if the buffer size is too large, buffers allocated during periods of peak activity may be rarely used while consuming memory that would otherwise be available for other tasks.

For request processing, EAServer uses 8k buffers by default, and uses 4k buffers only when a servlet calls ServletResponse.setBufferSize() to request a buffer size other than 8k. If your application never or seldom changes the buffer size, you can set com.sybase.jaguar.server.servlet.max4kbuffers to 0 so that 4k buffers are not pooled.

To access whether the settings are correct, examine the servlet request patterns to see if the number of concurrent requests often exceeds the buffer pool sizes. If so, consider increasing the value.

See also

com.sybase.jaguar.server.servlet.max8kbuffers

com.sybase.jaguar.server.servlet.max8kbuffers

Description

Specifies the number of internal 8K servlet response buffers.

Syntax

Same as for com.sybase.jaguar.server.servlet.max4kbuffers.

com.sybase.jaguar.server.servlet.mime-mapping

Description

Specifies MIME mappings for static files served outside of Web applications.

Syntax

mapping1, *mapping2*, ...

Where *mapping1*, *mapping2*, and so forth are mappings of the form:

(description=*desc*,mime-type=*mime-def*,extension=*ext*)

Where:

desc	Is an optional description.
mime-def	Is the MIME type specification, for example, `text/plain` or `text/sgml`.
ext	Is the file extension for files of this type.

Usage

A file's MIME type specifies how a server or browser should interpret the file. For example, whether the file contains plain text, formatted HTML, an image, or a sound recording. In a Web server, MIME mappings specify how a static file should be interpreted by mapping file extensions to MIME types. MIME mappings affect only static files. Servlets and JSPs must be coded to specify a MIME type for their response.

For more information on MIME types, visit:

```
http://www.oac.uci.edu/indiv/ehood/MIME/MIME.html
```

EAServer includes preconfigured MIME mappings. These settings override EAServer's preconfigured mappings.

See also

com.sybase.jaguar.webapplication.mime-mapping

com.sybase.jaguar.server.servlet.serverCheckPeerIPForHttpSession

Description

Specifies whether EAServer binds the client IP address to each servlet authenticated session and validates subsequent requests to insure the IP address is the same.

Syntax

`true` or `false`. The default is `false`.

Usage

A value of `true` can further protect against replay attacks against secure Web applications.

See also

Chapter 3, "Using Web Application Security," in the *EAServer Security Administration and Programming Guide*.

com.sybase.jaguar.server.servlet.servlet-mapping

Description

Associates request paths with servlets installed directly in this server.

Syntax

mapping1,*mapping2*,...

Where *mapping1*, *mapping2*, and so forth are mappings of the form:

(description=*desc*,url-pattern=*pattern*,servlet-name=*servlet*)

Where:

Variable	Specifies
desc	An optional text description of the mapping.
pattern	A URL pattern that can contain wildcards.
servlet	A servlet that is installed in this application. Servlets in the Web application have names formatted as *webapp*/*servlet*. Use only the *servlet* part of the servlet name.

Usage — This property affects servlets installed directly in the server; that is, those specified by the com.sybase.jaguar.server.servlets property. It does not affect servlets installed in a Web application.

See also — com.sybase.jaguar.server.servlets, com.sybase.jaguar.webapplication.servlet-mapping

com.sybase.jaguar.server.servlet.session-config

Description — Configures HTTP session properties for servlets installed directly in this server.

Syntax — Same as the Web application property com.sybase.jaguar.webapplication.session-config.

Usage — This property affects servlets installed directly in the server; that is, those specified by the com.sybase.jaguar.server.servlets property. It does not affect servlets installed in a Web application.

See also — com.sybase.jaguar.server.servlets, com.sybase.jaguar.webapplication.session-config

com.sybase.jaguar.server.servlet.trace

Description — Enables and disables trace logging in the EAServer servlet execution engine.

Syntax — `true` or `false`. The default is `false`, which disables tracing.

com.sybase.jaguar.server.servlet.welcome-file-list

Description	Configures welcome files for Web directories served outside of Web applications. Welcome files are used to satisfy HTTP requests that end in a directory name, rather than specifying the full path to a file or a path that is mapped to a servlet invocation.
Syntax	A comma-separated list of file names which cannot contain path separators.
See also	com.sybase.jaguar.webapplication.welcome-file-list

com.sybase.jaguar.server.servlets

Description	Specifies the servlets that are installed directly in this server and therefore part of the default Web application *EASDefault*.
Syntax	A comma-separated list of servlet names that are defined in the repository.
Usage	Servlets can also be installed as part of a user-defined Web application. Web applications are the preferred deployment mechanism where portability to other J2EE application servers is a concern.
See also	com.sybase.jaguar.server.webapplications, com.sybase.jaguar.server.applications, Servlet properties on page 563

com.sybase.jaguar.server.stacksize

Description	On UNIX platforms, configures the thread stack size.
Syntax	The stack size in bytes. If not set, the default is 256K.
See also	“Configuring server stack size” on page 68

com.sybase.jaguar.server.timeout

Description	Specifies the default instance timeout for stateful components running in the server.
Syntax	The default is 0.

com.sybase.jaguar.server.traceattentions

Description	Specifies whether TDS protocol attentions are being traced.
Syntax	`true` or `false`. The default is `false`, which disables tracing.

com.sybase.jaguar.server.tracenetdriver

Description	Specifies whether network driver requests are traced.
Syntax	`true` or `false`. The default is `false`, which disables tracing.

com.sybase.jaguar.server.tracenetrequests

Description	Specifies whether transport control layer requests are being traced. (The transport control layer is an internal EAServer library that acts as an interface to network drivers.)
Syntax	`true` or `false`. The default is `false`, which disables tracing.

com.sybase.jaguar.server.tracetdsdata

Description	Specifies whether TDS packet contents are traced.
Syntax	`true` or `false`. The default is `false`, which disables tracing.

com.sybase.jaguar.server.tracetdshdr

Description	Specifies whether TDS header contents are traced.
Syntax	`true` or `false`. The default is `false`, which disables tracing.

com.sybase.jaguar.server.truncatelog

Description	Obsolete since version 5.0. Use the log profile properties to specify whether the log should be truncated on start-up.
Syntax	N/A.
See also	Log profile properties on page 460

com.sybase.jaguar.server.tx_retry

Description	Specifies a server wide default for the component Automatic Transaction Retry property (com.sybase.jaguar.component.tx_retry).
Syntax	`true` or `false`. If not set, the default is `false`.
See also	com.sybase.jaguar.component.tx_retry

com.sybase.jaguar.server.tx_timeout

Description	Specifies the default transaction timeout for components running in the server.
Syntax	The timeout period, in seconds, with 0 indicating no timeout. The default is 0.
Usage	In EAServer Manager, set this property using the Advanced tab in the Server Properties dialog box.
See also	com.sybase.jaguar.component.tx_timeout

com.sybase.jaguar.server.TxManager.logfile

Description	Specifies the name of the transaction log file, which the recovery manager reads to perform transaction recovery.
Syntax	The file name. The default is *<serverName>Recovery.log*.
Usage	In EAServer Manager, set this property using the Advanced tab in the Server Properties dialog box.
See also	com.sybase.jaguar.server.TxManager.RecoveryEnabled, Chapter 2, “Understanding Transactions and Component Lifecycles,” in the *EAServer Programmer’s Guide*.

com.sybase.jaguar.server.TxManager.logsize

Description	Specifies the minimum size of the transaction log file.
Syntax	The file size. The default is 1MB.
Usage	In EAServer Manager, set this property using the Advanced tab in the Server Properties dialog box.
See also	com.sybase.jaguar.server.TxManager.logfile

com.sybase.jaguar.server.TxManager.RecoveryEnabled

Description	Specifies whether the recovery manager is enabled.
Syntax	`true` or `false`. The default is `false`, which disables the Recovery Manager.
Usage	In EAServer Manager, set this property using the Advanced tab in the Server Properties dialog box.
See also	com.sybase.jaguar.server.TxManager.logfile

com.sybase.jaguar.server.txmodel

Description: Specifies the transaction coordination model for the server.

Syntax

Value	To indicate
`JTS` (the default)	JTS/JTA
`dtc`	Microsoft DTC (Windows platforms only)

See also: "Transactions" on page 29

com.sybase.jaguar.server.unix.groupname

Description	On UNIX platforms, specifies a group name for the effective user ID of the server process.
Syntax	The operating system group name.
See also	com.sybase.jaguar.server.unix.username, "Changing the effective user ID of the server process" on page 54

com.sybase.jaguar.server.unix.username

Description	On UNIX platforms, specifies the effective user ID of the server process.
Syntax	The operating system user name.
See also	com.sybase.jaguar.server.unix.groupname, "Changing the effective user ID of the server process" on page 54

com.sybase.jaguar.server.validateusersgroups

Description	Specifies whether user membership in operating system groups is checked when determining a user's membership in EAServer roles.
Syntax	`true` or `false`. The default is `false`.

com.sybase.jaguar.server.webapplications

Description	Specifies the names of the Web applications that are installed in the server.
Syntax	A comma-separated list of the Web application names.
Usage	Web applications may be installed to a server directly using this property or as part of an application using the property com.sybase.jaguar.server.applications.
See also	com.sybase.jaguar.server.applications

com.sybase.jaguar.server.XSLTfactory

Description	Specifies the class name for a custom XSLT XML parser factory class.
Syntax	The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.server.java.classes property, and the file must be placed in either the EAServer *java/classes* or *java/lib* directory.
See also	com.sybase.jaguar.server.jagmgr.XSLTFactoryChoice, com.sybase.jaguar.server.DOMfactory, com.sybase.jaguar.server.SAXfactory

Environment variable properties

Description	You can set environment variables for the server process by specifying them as server properties. Values specified in environment variables overwrite any values set in the server start-up scripts. You can use this feature to set properties when a server is run as a Windows service. In this case, the server start-up script settings do not apply.
Syntax	Specify the environment variable name as the property name, and the value as the property value. For example: `MYPATH=/work/EAServer/deploy`

Usage

The following variables have default values that can be overwritten by specifying a property with a new value:

Variable	Default value
JAGUAR_HOST_NAME	The host name of the machine where the server is running. You can override the default if the machine supports multiple addresses (host names), and you want to force the address that is used when the listener configuration uses the ${JAGUAR_HOST_NAME} macro.
JAGUAR_IP_ADDRESS	The IP address of the machine where the server is running. You can override the default if the machine supports multiple addresses, and you want to force the address that is used when the listener configuration uses the ${JAGUAR_IP_ADDRESS} macro.
DLL_SUFFIX	The platform-specific suffix used by shared libraries or DLLs. Do not change the default value.
JAGUAR_PLATFORM	The platform-identifier used in generated C++ component make files, and to load platform-specific shared libraries in mixed-architecture clusters. Do not change the default value.

Servlet properties

Description

Servlet property names begin with com.sybase.jaguar.servlet. Servlet properties apply to servlet entities defined in the EAServer installed servlets folder (com.sybase.jaguar.server.servlets property), as well as to servlets and JSPs installed in a Web application (com.sybase.jaguar.webapplication.webcomponents property).

com.sybase.jaguar.servlet.cache

Description

Enables and disables servlet response caching.

Syntax

To enable caching:

CtsComponents/PageCache

No value means caching is disabled.

com.sybase.jaguar.servlet.cache.entire-tree

Description	Specifies whether to cache all the pages or files that are invoked by a Web component.
Syntax	`true` or `false`. The default is `false`, which means that a page is cached only when a client requests it directly.
See also	Chapter 5, "Web Application Tuning," in the *EAServer Performance and Tuning Guide*.

com.sybase.jaguar.servlet.cache.locale-sensitive

Description	When response caching is enabled, specifies whether to include the accept-languages header in the cache key.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.servlet.cache

com.sybase.jaguar.servlet.cache.message-topics

Description	When response caching is enabled, specifies message service topic names used to synchronize the cache with an external storage mechanism such as a database.
Syntax	A comma-separated list of topic names.
See also	com.sybase.jaguar.servlet.cache

com.sybase.jaguar.servlet.cache.request-headers

Description	When response caching is enabled, specifies request headers to include in the cache key.
Syntax	A comma-separated list of request headers. For example, if you include `date`, EAServer looks for cache entries whose date headers match the request's date header.
See also	com.sybase.jaguar.servlet.cache

com.sybase.jaguar.servlet.cache.request-parameters

Description	When response caching is enabled, specifies request parameters to include in the cache key.
Syntax	A comma-separated list of parameter names, or "*" to include all of them.
See also	com.sybase.jaguar.servlet.cache

com.sybase.jaguar.servlet.cache.session-attributes

Description	When response caching is enabled, specifies session attributes to include in the cache key.
Syntax	A comma-separated list of attribute names, or "*" to include all of them.
See also	com.sybase.jaguar.servlet.cache

com.sybase.jaguar.servlet.cache.timeout

Description	When response caching is enabled, specifies the cache timeout value.
Syntax	The timeout value in seconds. A value of 0 indicates infinity.
See also	com.sybase.jaguar.servlet.cache

com.sybase.jaguar.servlet.cache.use-sessionid

Description	When response caching is enabled, specifies whether to include the session ID in the cache key.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.servlet.cache

com.sybase.jaguar.servlet.description

Description	An optional description of the servlet or JSP.
Syntax	The descriptive text.

com.sybase.jaguar.servlet.destroy.wait-time

Description	Overrides the server or Web application servlet destroy timeout setting for this servlet.
Syntax	Same as the server property com.sybase.jaguar.server.servlet.destroy-wait-time. For servlets installed in a Web application, the default is the value of the Web application property com.sybase.jaguar.webapplication.destroy-wait-time. For servlets installed directly in the server, the default is the value of the server property com.sybase.jaguar.server.servlet.destroy-wait-time.
See also	com.sybase.jaguar.servlet.init.timeout, com.sybase.jaguar.server.servlet.destroy-wait-time, com.sybase.jaguar.webapplication.destroy-wait-time

com.sybase.jaguar.servlet.files

Description	Specifies additional files to be included if this servlet is archived or replicated to another installation using the synchronize feature.
Syntax	Same as for the com.sybase.jaguar.applicationclient.files component property.
See also	com.sybase.jaguar.component.files, com.sybase.jaguar.webapplication.files

com.sybase.jaguar.servlet.init-param

Description	Specifies initialization parameters for the servlet.
Syntax	Same as for the com.sybase.jaguar.filter.init-param filter property.
See also	com.sybase.jaguar.filter.init-param

com.sybase.jaguar.servlet.init.timeout

Description	Specifies how long the server should wait for the servlet's init method to return.
Syntax	A value from Table B-21. If this property is not set for a servlet in a Web application, the default is the value of the Web application property com.sybase.jaguar.webapplication.init-timeout. For servlets installed directly in the server, the default is the value of the server property com.sybase.jaguar.server.servlet.init-timeout.

Table B-21: Initialization timeout values

Value	To indicate
-1	(The server or Web application default.) init can run indefinitely, unless the server is shutdown or refreshed. If the init method is still running when the server is shutdown or refreshed, the server does not wait for init to complete before shutting down or refreshing the servlet.
0	init can run indefinitely. Sybase does not recommend this setting, because deadlocks or other hangs in the init method can cause the server to hang when shutting down or refreshing the servlet.
A positive integer.	The number of seconds to wait for init to return. If the init method is still running when the server is shutdown or refreshed, the server waits the specified time for init to return.

See also com.sybase.jaguar.server.servlet.init-timeout, com.sybase.jaguar.webapplication.init-timeout

com.sybase.jaguar.servlet.javacache.enabled

Description Enables the servlet Java cache mechanism for this servlet or JSP.

Syntax `true` or `false`. The default is `false`. A value of `true` enables the Java cache. If both page caching and Java caching are enabled, Java caching is used. (The com.sybase.jaguar.servlet.cache property enables page caching.)

Usage When enabled, this feature allows caching of servlet output in Java core memory, which offers a faster response than page caching for servlets that can run under these limitations:

- The output does not change during the cache timeout period, and does not depend on request method, parameters, or headers.
- The servlet runs with exact path mappings. Responses are not cached if the servlet uses prefix mappings, default mappings, or extension mappings.
- Cached content is returned without modification.
- Cached response headers are returned without modification, except for:
  - The Set-Cookie header, which depends on the com.sybase.jaguar.servlet.javacache.session property.
  - The Connection header, which depends on the request Connection header.

- Return code 200 is used for all cached replies. If the original response uses any other return code, it is not cached.
- Chunked replies are not cached.
- Only responses to GET requests are cached.

Servlets that cannot run under these limitations can still use page caching, enabled by setting com.sybase.jaguar.servlet.cache.

See also com.sybase.jaguar.servlet.cache, com.sybase.jaguar.servlet.javacache.maxsize, com.sybase.jaguar.servlet.javacache.session, com.sybase.jaguar.servlet.javacache.timeout

com.sybase.jaguar.servlet.javacache.maxsize

Description When using the Java cache mechanism, specifies the size, in kilobytes, of the largest reply that can be cached. Responses larger than this are not cached.

Syntax The default is 8. The hard upper limit is 100. If you set a value greater than 100, the effective cache size is 100K.

See also com.sybase.jaguar.servlet.javacache.enabled, com.sybase.jaguar.servlet.javacache.session, com.sybase.jaguar.servlet.javacache.timeout

com.sybase.jaguar.servlet.javacache.session

Description When using the Java cache mechanism, specifies how session cookie settings are treated in response headers when using the Java cache mechanism.

Syntax Allowable values are:

Value	To indicate
`no` (the default)	No session support. EAServer does not check request Set-Cookie headers or return response Set-Cookie headers.
`keep`	Attempt to preserve valid sessions. If the request includes a session identifier, EAServer checks if the session is valid. If it is not, the response Set-Cookie header is set to indicate an invalid session. Otherwise, no Set-Cookie header is returned.

Value	To indicate
`create`	Preserve valid session and create a new session if the previous session is invalid. If the request includes a session identifier, EAServer checks if the session is valid. If it is not, the response Set-Cookie header is set to indicate a new session. Otherwise, no Set-Cookie header is returned.

See also com.sybase.jaguar.servlet.javacache.enabled, com.sybase.jaguar.servlet.javacache.maxsize, com.sybase.jaguar.servlet.javacache.timeout

com.sybase.jaguar.servlet.javacache.timeout

Description When using the Java cache mechanism, specifies the time, in seconds, that cached responses remain valid.

Syntax The timeout value in seconds. The default is 60. A negative value specifies an infinite timeout, that is, cached responses do not expire.

See also com.sybase.jaguar.servlet.javacache.enabled, com.sybase.jaguar.servlet.javacache.maxsize, com.sybase.jaguar.servlet.javacache.session

com.sybase.jaguar.servlet.java.class

Description Specifies the servlet implementation class.

Syntax The Java class name, for example:

```
com.acme.SearchServlet
```

See also com.sybase.jaguar.servlet.java.classes

com.sybase.jaguar.servlet.java.classes

Description Specifies additional classes to be reloaded when the servlet is refreshed.

Syntax Same as for the com.sybase.jaguar.application.java.classes component property.

Usage If configuring servlets or JSPs that are installed in a Web application, see "Custom class lists for Web applications" in Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*.

If configuring servlets installed directly in the server's Installed Servlets folder, see "Custom class lists for servlets installed directly in the server" in Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*.

See also: com.sybase.jaguar.servlet.java.class, com.sybase.jaguar.webapplication.java.classes

com.sybase.jaguar.servlet.jsp.compile-extra-cp

Description: Specifies additional JAR files and directories to include in the JSP compiler class path.

Syntax: Same as for com.sybase.jaguar.webapplication.jsp.compile-extra-cp

com.sybase.jaguar.servlet.jsp.compile-use-eas-cp

Description: Applies to JSPs only. Specifies whether the EAServer process CLASSPATH should be included in the compilation class path when compiling this JSP.

Syntax: `true` or `false`. The default of `true` indicates that the server class path should be included. Set this property to `false` to exclude entries from the EAServer process CLASSPATH setting in the JSP compiler class path.

See also: com.sybase.jaguar.webapplication.jsp.compile-use-eas-cp

com.sybase.jaguar.servlet.jsp.compile-use-third-party

Description: Applies to JSPs only. Specifies whether JAR files in the EAServer *java/lib* directory are automatically be included in the class path when compiling this JSP.

Syntax: `true` or `false`. If this property is `true`, all JAR files in this directory are included. The default is `false`. This property is ignored if the com.sybase.jaguar.webapplication.jsp.compile-use-eas-cp property is set to `true` for the Web application or the com.sybase.jaguar.servlet.jsp.compile-use-eas-cp is set to `true` for the JSP.

See also: com.sybase.jaguar.webapplication.jsp.compile-use-third-party

com.sybase.jaguar.servlet.jsp-file

Description: If this servlet is generated from a JSP file, specifies the *.jsp* file name.

Syntax: The file name, including the path relative to the Web application's context root directory. For example:

```
annotation/annotated_account.jsp
```

See also: com.sybase.jaguar.servlet.servletorjsp

com.sybase.jaguar.servlet.large-icon

Description: Specifies the name of the large icon file associated with the servlet. This property is not used in EAServer, but accommodated to comply with the Servlet 2.3 Web archive descriptor.

Syntax: A file name.

com.sybase.jaguar.servlet.load-on-startup

Description: Specifies whether the servlet is loaded at server start-up time, or after the first client request.

Syntax

Value	To specify
No value.	The servlet is not loaded at start-up.
0	Load at start-up, with unspecified position relative to other servlets loaded at start-up.
A positive integer.	Load at start-up, with specified position relative to other servlets loaded at start-up.

com.sybase.jaguar.servlet.name

Description: The servlet name.

Syntax: For a servlet in a Web application:

web-app/*servlet*

For a servlet not in a Web application:

servlet

Where *servlet* is the servlet name as displayed in EAServer Manager, and *web-app* is the Web application name.

com.sybase.jaguar.servlet.security.runasidentity

Description

Specifies the run-as identity used for component calls. If this property is not set, intercomponent calls use the client identity.

Syntax

(specified=*id*,role=*role-name*,desc=*desc*)

where:

Variable	Specifies
id	A logical identity name which must be mapped to an EAServer identity by setting a corresponding package property, com.sybase.jaguar.webapplication.runasidentity.<id> where `<id>` is the logical identity name.
role-name	A logical role name which must be mapped to an EAServer role by setting a corresponding Web application property, com.sybase.jaguar.webapplication.security-role.<j2ee-role> where `<j2ee-role>` is the logical role name. The mapping can also be established by setting the application property com.sybase.jaguar.application.security-role.<j2ee-role>. The mapped EAServer identity should be in the mapped EAServer role.
desc	An optional description of the run-as authorization requirement. The description can help users when the Web application is deployed to another server, and the deployer must choose a different identity mapping.

See also

com.sybase.jaguar.webapplication.runasidentity.<id>, com.sybase.jaguar.webapplication.security-role.<j2ee-role>, com.sybase.jaguar.application.security-role.<j2ee-role>

com.sybase.jaguar.servlet.servletorjsp

Description

Specifies whether this servlet is generated from a JSP.

Syntax

Value	To indicate
`SERVLET`	A servlet
`JSP`	A JSP

See also	com.sybase.jaguar.servlet.jsp-file

com.sybase.jaguar.servlet.session.allowed

Description	For servlets not in a Web application, specifies whether the servlet can use sessions.
Syntax	`true` or `false`. The default is `false`.
Usage	For Web application servlets, sessions are always enabled.

com.sybase.jaguar.servlet.session.timeout

Description	For servlets not in a Web application, specifies the session timeout.
Syntax	The timeout value in seconds. A value of 0 indicates infinity.
Usage	For Web application servlets, the Web application com.sybase.jaguar.webapplication.session-config property specifies the session timeout value.
See also	com.sybase.jaguar.webapplication.session-config

com.sybase.jaguar.servlet.singlethread

Description	Specifies whether an instance of the servlet class can be run simultaneously on multiple threads.
Syntax	`true` or `false`. The default is `false`.
See also	com.sybase.jaguar.servlet.singlethread.poolsize

com.sybase.jaguar.servlet.singlethread.poolsize

Description	If the servlet is single-threaded, specifies the number of threads to run the servlet on. More threads may decrease the average client response time by eliminating the need to serialize requests.
Syntax	A positive integer, or 0 to indicate no limit. The default is 1.
See also	com.sybase.jaguar.servlet.singlethread

com.sybase.jaguar.servlet.small-icon

Description — Specifies the name of the small icon file associated with the servlet. This property is not used in EAServer, but accommodated to comply with the Servlet 2.3 Web archive descriptor.

Syntax — A file name.

Thread monitor properties

Description — Thread monitors provide a means to limit the execution time devoted to specified components and component methods. You can assign components and methods to a thread monitor to ensure that no more than a specified maximum number of threads will be active at any point executing the methods and components assigned to the monitor.

You can also use thread monitors without a limit on the number of threads. Doing so allows you to use the monitor trace properties to record performance data.

Thread monitors are not active for EJB inter-component calls that use the lightweight EJB container.

Usage — To create a thread monitor, create a property file in the *Repository/ThreadMonitor* directory of your installation. Create this directory if it does not exist. For example, to create a monitor named "MyMonitor," create the file *Repository/ThreadMonitor/MyMonitor.props*.

In the properties file, add settings for the thread monitor properties listed in Table B-22. Actual property names are prefixed with "com.sybase.jaguar.threadmonitor." For example:

```
com.sybase.jaguar.threadmonitor.name=MyMonitor
com.sybase.jaguar.threadmonitor.maxthreads=0
com.sybase.jaguar.threadmonitor.nested=false
com.sybase.jaguar.threadmonitor.trace=false
com.sybase.jaguar.threadmonitor.callstats=0
```

Table B-22: Thread monitor properties

Property	Legal values	Default value	Description
name	The entity name	N/A	Name of the thread monitor.

Property	Legal values	Default value	Description
maxthreads	0 or positive	0	Maximum number of threads permitted to be simultaneously active under this monitor. Zero indicates no maximum, in which case trace and callstats may be useful for performance tuning.
nested	true/false	false	Enables this monitor as nested. A nested monitor can limit thread activity on a thread that is already controlled by a monitor. Nested monitors must be used with care, as they can result in deadlock among EAServer threads, requiring server restart.
trace	true/false	false	Enables tracing to indicate the maximum (peak) number of active and waiting threads in the server log. Also enables call statistics to be printed if callstats is nonzero.
callstats	0 or positive	0	Number of remote method invocations between call statistics trace entries that are logged for this monitor. A value of 0 indicates no call statistics are logged. A value of 1 indicates call statistics are recorded for every remote method call.

Note Cluster synchronization does not yet include the property files for thread monitors.

To assign a component to a thread monitory, set the component property com.sybase.jaguar.component.monitor to the name of the thread monitor.

To assign a component to a thread monitory, set the method property com.sybase.jaguar.method.monitor to the name of the thread monitor.

Regenerate and recompile the component skeleton after you assign the component or method to a monitor.

See also com.sybase.jaguar.component.monitor, com.sybase.jaguar.method.monitor

Web application properties

Description Web application property names begin with `com.sybase.jaguar.webapplication`.

com.sybase.jaguar.webapplication.application

Description The application where this Web application is installed.

Syntax	The application name, or no value if the Web application is not installed in any application.
Usage	A Web application can be installed in one application, or directly to one or more servers. A Web application cannot be installed both directly in a server and to an application.
See also	com.sybase.jaguar.application.webapplications, com.sybase.jaguar.server.webapplications

com.sybase.jaguar.webapplication.cache.locale-sensitive

Description	For installed servlets, specifies the default for the servlet property com.sybase.jaguar.servlet.cache.locale-sensitive.
Syntax	Same as com.sybase.jaguar.servlet.cache.locale-sensitive.

com.sybase.jaguar.webapplication.cache.message-topics

Description	For installed servlets, specifies the default for the servlet property com.sybase.jaguar.servlet.cache.message-topics.
Syntax	Same as com.sybase.jaguar.servlet.cache.message-topics.

com.sybase.jaguar.webapplication.cache.request-headers

Description	For installed servlets, specifies the default for the servlet property com.sybase.jaguar.servlet.cache.request-headers.
Syntax	Same as com.sybase.jaguar.servlet.cache.request-headers.

com.sybase.jaguar.webapplication.cache.request-parameters

Description	For installed servlets, specifies the default for the servlet property com.sybase.jaguar.servlet.cache.request-parameters.
Syntax	Same as com.sybase.jaguar.servlet.cache.request-parameters.

com.sybase.jaguar.webapplication.cache.session-attributes

Description — For installed servlets, specifies the default for the servlet property com.sybase.jaguar.servlet.cache.session-attributes.

Syntax — Same as com.sybase.jaguar.servlet.cache.session-attributes.

com.sybase.jaguar.webapplication.cache.timeout

Description — For installed servlets, specifies the default for the servlet property com.sybase.jaguar.servlet.cache.timeout.

Syntax — Same as com.sybase.jaguar.servlet.cache.timeout.

com.sybase.jaguar.webapplication.cache.use-sessionid

Description — For installed servlets, specifies the default for the servlet property com.sybase.jaguar.servlet.cache.use-sessionid.

Syntax — Same as com.sybase.jaguar.servlet.cache.use-sessionid.

com.sybase.jaguar.webapplication.charset.inputdata

Description — Specifies the character set for JSP or servlet request body data (retrieved with ServletRequest.getReader or ServletRequest.getInputStream).

Syntax — A list of URL-pattern and Java character set name pairs. Use this syntax, where *URL_pattern* is the URL pattern to which the character set applies, and *character_set* is the name of the Java character set:

```
(url-pattern=URL_pattern,charset=character_set),
(url-pattern=URL_pattern,charset=character_set)
```

For example, for a Web application with two directories, */en* and */ko*, in its document root where all files under */en* are 8859_1 encoded and all files under */ko* are KSC5601 encoded, specify the character sets like this:

```
(url-pattern=/en/*,charset=8859_1),
(url-pattern=/ko/*,charset=KSC5601)
```

The server's default character set is used for URL patterns that are not specified. If you specify a character set that is not supported, it is not added to the mapping and the server's default character set is used.

Usage	This property is not supported for the default Web application.
See also	com.sybase.jaguar.webapplication.charset.inputparam, com.sybase.jaguar.webapplication.charset.jspcompile, Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.webapplication.charset.inputparam

Description	Specifies the character set for servlet and JSP request parameters.
Syntax	Same as com.sybase.jaguar.webapplication.charset.inputdata.
Usage	This property is not supported for the default Web application.
See also	com.sybase.jaguar.webapplication.charset.inputdata, com.sybase.jaguar.webapplication.charset.jspcompile,, Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.webapplication.charset.jspcompile

Description	Specifies the character set for JSP compilation.
Syntax	Same as com.sybase.jaguar.webapplication.charset.inputdata.
Usage	This property is not supported for the default Web application.
See also	com.sybase.jaguar.webapplication.charset.inputdata, com.sybase.jaguar.webapplication.charset.inputparam,, Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.webapplication.classloaderpolicy

Description	Specifies how the custom class loader (version 2) resolves version conflicts when you specify the same class at multiple levels in the class loader hierarchy.
Syntax	Same as com.sybase.jaguar.application.classloaderpolicy.
See also	com.sybase.jaguar.server.jvm.classloader, com.sybase.jaguar.application.classloaderpolicy, com.sybase.jaguar.component.classloaderpolicy, com.sybase.jaguar.package.classloaderpolicy

Chapter 30, “Configuring Custom Java Class Lists,” in the *EAServer Programmer’s Guide*.

com.sybase.jaguar.webapplication.context-param

Description	Specifies the context initialization parameters for the servlets in the Web application.
Syntax	Same as the servlet property com.sybase.jaguar.server.servlet.context-param.
Usage	All servlets and JSPs in a Web application share a common set of context initialization properties specified by the deployment descriptor. Servlet code can retrieve the values by calling the getInitParamers() and getInitParameterNames() methods in interface javax.Servlet.ServletContext. Environment properties can be used for the same purpose as context-initialization properties, and allow additional datatypes besides java.lang.String. See com.sybase.jaguar.server.servlet.context-param on page 552 for an explanation of context parameters.
See also	com.sybase.jaguar.webapplication.env-entry

com.sybase.jaguar.webapplication.context-path

Description	The request-path prefix that clients use in URLs to access your Web application’s static content, servlets, and JSPs.
Syntax	The context path, which must be a string containing no path separators. For example, if you enter “estore,” users access your Web application with the prefix: http://*host*:*port*/estore/ The default context path is the name of the Web application.

com.sybase.jaguar.webapplication.cookie.persistent

Description	Specifies whether session data cookies are persistent or temporary.
Syntax	`true` or `false`. The default is `false`, which means cookies are temporary.

Usage — When this property is false (the default), EAServer stores session data in temporary cookies. When set to true, EAServer sends a persistent cookie that expires when the Web application session-timeout setting expires. This property affects only the cookies that EAServer creates to store session data for the Web application (available to servlets and JSPs via request.getSession). It does not affect cookies created explicitly by servlets and JSPs.

See also — com.sybase.jaguar.webapplication.session-config

com.sybase.jaguar.webapplication.dependencies

Description — Specifies dependencies on standard Java extensions.

Syntax — A string of the form:

dep1, *dep2*, *dep3*, ...

Where *dep1*, *dep1*, *dep3* are of the form:

(prefix=*name*,extension-name=*name*,specification-version=*vnum*,specification-vendor=*spec-vendor*,implementation-version=*inum*,implementation-vendor-id=*impl-vendor-id*,implementation-vendor=*impl-vendor*,implementation-url=*impl-url*)

Table B-23 describes the values and the corresponding entries in the *manifest.mf* file within an extension JAR file.

Table B-23: Java extension properties

Property value	Manifest entry	Description
name	Extension-Name	The extension name.
vnum	Specification-Version	The version number of the specification that the extension conforms to.
spec-vendor	Specification-Vendor	The company or organization responsible for the specification that the extension conforms to.
inum	Implementation-Version	The implementation version number.
impl-vendor	Implementation-Vendor	The company or organization responsible for the implementation.
impl-vendor-id	Implementation-Vendor-ID	A unique identifier for the company or organization responsible for the implementation. Usually follows the reverse-domain naming convention used in Java packages, for example, “com.sybase.”
impl-url	Implementation-URL	A Web URL to obtain information on the implementation.

com.sybase.jaguar.webapplication.default.protectedpage

Description	Specifies a default page to redirect to if a servlet uses direct form login to authenticate the user.
Syntax	The page URL. You can override the setting in the servlet session properties before submitting the direct form login request. If no page is specified in the session or Web application properties, EAServer redirects the user to the Web application's welcome page.
Usage	Direct form login allows you to authenticate the user without requiring them to visit the form login page. For more information, see "Web application direct form login" in Chapter 3, "Using Web Application Security," in the *EAServer Security Administration and Programming Guide*.

com.sybase.jaguar.webapplication.destroy-wait-time

Description	The number of seconds that EAServer should wait for servlet service calls to return before calling the servlet destroy method. Affects all servlets installed in the Web application.
Syntax	The timeout value in seconds. A value of 0 indicates infinity.
See also	com.sybase.jaguar.webapplication.init-timeout

com.sybase.jaguar.webapplication.distributable

Description	Specifies whether multiple instances of the Web application can run in a distributed server environment on different servers.
Syntax	`true` or `false`. The default is `false`.
Usage	Setting this property to true causes Web client session data to be stored in a remote database, so all servers in a cluster can share the data. Additional configuration is required to configure the data store. See Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide* for more information.
See also	com.sybase.jaguar.webapplication.distribute.type

com.sybase.jaguar.webapplication.distribute.type

Description — Specifies how EAServer replicates HTTP session data for distributable Web applications.

Syntax — Allowable values are:

Value	To indicate
`database`	The default EAServer replicates session data using persistent storage provided by a remote database.
`inmemory`	EAServer replicates session data in-memory. See Chapter 21, "Creating Web Applications," in the *EAServer Programmer's Guide* for more information.

See also — com.sybase.jaguar.webapplication.distributable

com.sybase.jaguar.webapplication.DOMfactory

Description — Specifies the class name for a custom DOM XML parser factory class.

Syntax — The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.webapplication.java.classes property. The file must be placed in one of the following directories; EAServer searches the directories in the given order:

1 The *WEB-INF/lib* directory under the Web application's context root

2 *$JAGUAR/java/classes*

3 *$JAGUAR/java/lib*

See also — com.sybase.jaguar.webapplication.jagmgr.DOMFactoryChoice, com.sybase.jaguar.webapplication.SAXfactory, com.sybase.jaguar.server.XSLTfactory

com.sybase.jaguar.webapplication.ejb-local-ref

Description — Specifies a list of EJB local references that define aliased JNDI names for local EJB components invoked by servlets in the Web application.

Syntax — *ejb-ref1*, *ejb-ref2*, ...

Where *ejb-ref1*, *ejb-ref2*, and so forth follow the syntax of EJB local reference properties on page 447.

Usage	Use this property for beans invoked through the local interface. Use com.sybase.jaguar.webapplication.ejb-ref for beans invoked through the remote interface.
See also	EJB local reference properties on page 447, com.sybase.jaguar.webapplication.ejb-ref

com.sybase.jaguar.webapplication.ejb-ref

Description	Specifies a list of EJB references that define aliased JNDI names for EJB components invoked by servlets in the Web application.
Syntax	*ejb-ref1*, *ejb-ref2*, ... Where *ejb-ref1*, *ejb-ref2*, and so forth follow the syntax of EJB reference properties on page 447.
Usage	Use this property for beans invoked through the remote interface. Use com.sybase.jaguar.webapplication.ejb-local-ref for beans invoked through the local interface.
See also	EJB reference properties on page 447, com.sybase.jaguar.webapplication.ejb-local-ref

com.sybase.jaguar.webapplication.env-entry

Description	Environment properties allow you to specify global read-only data for use by the servlets in the Web application. Servlets must use JNDI to retrieve environment properties, using the prefix `java:comp/env` in JNDI lookups.
Syntax	See Environment properties on page 452.

com.sybase.jaguar.webapplication.files

Description	Specifies additional files to be included when the Web application is exported into a Jaguar JAR file or replicated using the synchronization feature.
Syntax	Same as com.sybase.jaguar.applicationclient.files.

com.sybase.jaguar.webapplication.filter-mapping

Description — Associates filters with servlets and URL paths.

Syntax — *mapping1*, *mapping2*, *mapping3*, ...

Where *mapping1*, *mapping2*, *mapping3* are strings of the form:

(description=*desc*,filter-name=*filter*,*type=pattern-or-servlet*)

Where:

- *desc* is an optional description of the mapping.
- *filter* is the filter name.
- *type* is `url-pattern` or `servlet-name`.
- *pattern-or-servlet* is the servlet name (if type is servlet-name) or URL pattern (if type is `url-pattern`).

See also — com.sybase.jaguar.webapplication.filters

com.sybase.jaguar.webapplication.filters

Description — Specifies filters installed in this Web application.

Syntax — A comma-separated list of filter names.

See also — Filter properties on page 453, com.sybase.jaguar.webapplication.filters

com.sybase.jaguar.webapplication.get-serverinfo-from

Description — When using a Web server redirector, configures the source for information returned by the HTTPServletRequest methods getScheme, getServerPort, and getServerName.

Syntax — A value from the following table:

Value	Specifies
`source`	Return the server, host, and scheme (protocol) for the Web server that is running the redirector.
`server`	Return the server, host, and scheme (protocol) for the EAServer listener to which the redirector connects.

Value	Specifies
`proxy`	Return the scheme (protocol), host, and port from the HTTP proxy settings on the HTTP Config tab in Server Properties. If these settings are not present, use the EAServer listener values.

com.sybase.jaguar.webapplication.httpdomain.override

Description: If set to true, the server HTTP Domain Name property (com.sybase.jaguar.server.http.domainname) is ignored for this Web application.

Syntax: `true` or `false`. The default is `false`.

See also: com.sybase.jaguar.server.http.domainname, "Configuring redirection addresses when using a proxy server" on page 27

com.sybase.jaguar.webapplication.init-timeout

Description: Specifies how long to wait for each installed servlet's init method to return.

Syntax: Same as the servlet property com.sybase.jaguar.servlet.init.timeout. You can override the setting for individual servlets by setting this servlet property.

See also: com.sybase.jaguar.webapplication.destroy-wait-time, com.sybase.jaguar.servlet.init.timeout

com.sybase.jaguar.webapplication.jagmgr.DOMFactoryChoice

Description: Specifies the XSLT parser configuration displayed in EAServer Manager.

Syntax

Value	To indicate
0	No parse
1	Platform default
2	The parser specified by the com.sybase.jaguar.webapplication.DOMfactory property

See also: com.sybase.jaguar.webapplication.DOMfactory, com.sybase.jaguar.webapplication.jagmgr.SAXFactoryChoice, com.sybase.jaguar.webapplication.jagmgr.XSLTFactoryChoice

com.sybase.jaguar.webapplication.jagmgr.SAXFactoryChoice

Description: Specifies the SAX parser configuration displayed in EAServer Manager.

Syntax

Value	To indicate
0	No parser
1	Platform default
2	The parser specified by the com.sybase.jaguar.webapplication.SAXfactory property

See also: com.sybase.jaguar.webapplication.SAXfactory, com.sybase.jaguar.webapplication.jagmgr.DOMFactoryChoice, com.sybase.jaguar.webapplication.jagmgr.XSLTFactoryChoice

com.sybase.jaguar.webapplication.jagmgr.XSLTFactoryChoice

Description: Specifies the XSLT parser configuration displayed in EAServer Manager.

Syntax

Value	To indicate
0	No parser
1	Platform default
2	The parser specified by the com.sybase.jaguar.webapplication.XSLTfactory property

See also: com.sybase.jaguar.webapplication.XSLTfactory, com.sybase.jaguar.webapplication.jagmgr.DOMFactoryChoice, com.sybase.jaguar.webapplication.jagmgr.SAXFactoryChoice

com.sybase.jaguar.webapplication.jarlist

Description: Specifies the class loading order when classes are loaded from JAR files in the *WEB-INF/lib* directory under the Web application's context root.

Syntax: A comma-separated list of JAR files. JAR files not listed in this property are loaded in directory order; that is, the order that they are returned in a directory listing.

See also: com.sybase.jaguar.webapplication.java.classes

com.sybase.jaguar.webapplication.java.classes

Description	Specifies additional classes and JAR files to be loaded by the Web application's custom class loader, in addition to those deployed in the *WEB-INF/lib* and *WEB-INF/classes* directories.
Syntax	Same as for com.sybase.jaguar.application.java.classes.
Usage	See "Custom class lists for Web applications" in Chapter 30, "Configuring Custom Java Class Lists," in the *EAServer Programmer's Guide*. In EAServer Manager, set this property using the Java Classes tab in the Application Properties dialog box.
See also	com.sybase.jaguar.application.java.classes, com.sybase.jaguar.server.java.classes, com.sybase.jaguar.servlet.java.classes

com.sybase.jaguar.webapplication.jsp.compile-extra-cp

Description	Specifies additional JAR files and directories to include in the JSP compiler class path.
Syntax	Specify the paths in a comma separated list, with paths relative to the EAServer installation directory. For example, to include *$JAGUAR/java/lib/iaws.jar* and $*JAGUAR/java/classes/extra.jar* you would set the property to: `java/lib/iaws.jar,java/classes/extra.jar`
See also	com.sybase.jaguar.servlet.jsp.compile-extra-cp

com.sybase.jaguar.webapplication.jsp.compile-use-eas-cp

Description	Specifies whether the EAServer process CLASSPATH should be included in the compilation class path when compiling JSPs.
Syntax	`true` or `false`. The default of `true` indicates that the server class path should be included. Set this property to `false` to exclude entries from the EAServer process CLASSPATH setting in the JSP compiler class path.
See also	com.sybase.jaguar.servlet.jsp.compile-use-eas-cp

com.sybase.jaguar.webapplication.jsp.compile-use-third-party

Description — Specifies whether JAR files in the EAServer *java/lib* directory are automatically be included in the class path when compiling JSPs.

Syntax — `true` or `false`. If this property is `true`, all JAR files in this directory are included. The default is `false`. This property is ignored if the com.sybase.jaguar.webapplication.jsp.compile-use-third-party property is set to `true`.

See also — com.sybase.jaguar.servlet.jsp.compile-use-third-party

com.sybase.jaguar.webapplicaton.jspc-interval

Description — Determines if and when the JSP runtime checks whether a JSP is current by comparing the modification times of the class and source files.

Syntax — Integer values, with the following meaning:

- If set to a negative number, the JSP runtime never checks.
- If set to 0, the JSP runtime always checks.
- To specify the number of seconds before the next check, set the value to a number greater than 0. If a request comes in before the time expires, the JSP is not checked.

If not set, the default is 1, which is appropriate for development testing. In production servers where JSP source files do not change, you can set the value to -1 or any negative value for improved performance.

com.sybase.jaguar.webapplication.keepgenerated

Description — Enables preservation of Java source files that EAServer generates to create servlets when compiling the JSPs installed in this Web application.

Syntax — `true` or `false`. The default of `false` indicates that EAServer deletes generated Java source files after compiling them.

See also — "Source and class file locations" in Chapter 24, "Creating JavaServer Pages," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.webapplication.large-icon

Description	Specifies the name of the large icon file associated with the Web application. This property is not used in EAServer, but accommodated to comply with the Servlet 2.3 Web archive descriptor.
Syntax	A file name.

com.sybase.jaguar.webapplication.lazydistributedhttpsessionvalidation

Description	Enables and disables lazy verification for distributed HTTP sessions.
Syntax	`true` or `false`. The default is `false`, which disables lazy authentication.
Usage	This setting affects only distributed Web applications. In the default configuration, EAServer validates a client's HTTP session during each request. If you enable lazy validation, EAServer validates the session only when a servlet or JSP calls ServletRequest.getSession() or ServletRequest.getSession(boolean). Lazy validation can improve performance. However, enabling lazy authentication has the following side effects: • The last-accessed-time session attribute is set only when the servlet or JSP accesses the session. Consequently, the session may expire sooner than expected if the client accesses only static pages or servlets and JSPs that do not access the session data. • When the session is invalidated, the client is not assigned a new session until they request a page that requires a session. • The client's security credentials (if any) are available only to JSPs and servlets that are marked protected via the security constraints property. Other pages cannot retrieve the client's credentials—the ServletRequest.getUserPrincipal() method returns null even though the client is logged in.
See also	com.sybase.jaguar.webapplication.distributable, "Deploying Web applications to a cluster" on page 149.

com.sybase.jaguar.webapplication.listeners

Description	Specifies application life cycle event listeners installed in this Web application.

Syntax — A comma-separated list of listener class names. Listeners are notified in the order that they are listed.

com.sybase.jaguar.webapplication.login-config

Description — Configures login authentication for the Web application.

Syntax — (form-login-config=(form-login-page=*lpage*,form-error-page=*epage*),realm-name=*realm*,auth-method=*method*)

Where:

Variable	Specifies
lpage	When *method* is FORM, the login form page. Unused for other methods.
epage	When *method* is FORM, the login error page.
realm	The realm name. When using the BASIC authentication method, some browsers display the realm name when prompting for authentication credentials.
method	One of the following: • `NONE` – no authentication is available. • `FORM` – you supply HTML or JSP forms to collect the user credentials and respond to authentication errors. The server loads the login form when authentication is required. • `BASIC` – when authentication is required, the browser collects the user credentials and sends them to the server in the HTTP basic authentication header. • `CLIENT-CERT` – the client connects to the server using HTTPS and provides an SSL certificate that the server accepts and authenticates. "Web Application Security" in the *EAServer Security Administration and Programming Guide* explains these options in more detail and describes how to create login and error form pages.

See also — com.sybase.jaguar.webapplication.security-constraint

com.sybase.jaguar.webapplication.mime-mapping

Description — Configures MIME mappings for the Web application to augment or override the server's default MIME mappings.

Syntax — Same as the servlet property com.sybase.jaguar.server.servlet.mime-mapping.

com.sybase.jaguar.webapplication.name

Description	Specifies the Web application name.
Syntax	The name.

com.sybase.jaguar.webapplication.refresh

Description	Specifies whether the Web application can be refreshed.
Syntax	`true` or `false`. The default is `true`, which allows refresh. If set to `false`, refresh operations from EAServer Manager, jagtool, and the management API do not reload this Web application or its installed Web components.

com.sybase.jaguar.webapplication.resource-env-ref

Description	Resource environment references are logical names applied to objects administered by EAServer.
Syntax	See Resource environment reference properties on page 489.
Usage	In EAServer Manager, set this property using the Resource Env Refs tab in the Web Application Properties dialog box.
See also	Resource environment reference properties on page 489.

com.sybase.jaguar.webapplication.resource-ref

Description	Specifies aliased JNDI names for database connections, JavaMail sessions, and URL factories used by the Web application.
Syntax	See Resource reference properties on page 496.
Usage	In EAServer Manager, set this property using the Resource Refs tab in the Web Application Properties dialog box.
See also	Resource reference properties.

com.sybase.jaguar.webapplication.runasidentity.<id>

Description	Maps an identity name used in servlet properties to an identity defined in the EAServer repository.

Syntax

Specify the identity referenced in a servlet com.sybase.jaguar.servlet.security.runasidentity property in the property name, for example:

```
com.sybase.jaguar.servlet.runasidentity.ddFooId
```

Specify the mapped EAServer identity as the value, for example:

```
fooId
```

See also

Security properties on page 500, com.sybase.jaguar.servlet.security.runasidentity

com.sybase.jaguar.webapplication.SAXfactory

Description

Specifies the class name for a custom SAX XML parser factory class.

Syntax

The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.webapplication.java.classes property. The file must be placed in one of the following directories; EAServer searches the directories in the given order:

1 The *WEB-INF/lib* directory under the Web application's context root

2 *$JAGUAR/java/classes*

3 *$JAGUAR/java/lib*

See also

com.sybase.jaguar.webapplication.jagmgr.SAXFactoryChoice, com.sybase.jaguar.webapplication.DOMfactory, com.sybase.jaguar.webapplication.XSLTfactory

com.sybase.jaguar.webapplication.sectrace

Description

Enables and disables security implementation tracing.

Syntax

`true` or `false`. The default is `false`.

com.sybase.jaguar.webapplication.security-constraint

Description

Associates required user roles and transport security for Web resource collections defined in the com.sybase.jaguar.webapplication.web-resource-collection property.

Syntax

constraint1,*constraint2*, ...

Where *constraint1*, *constraint2*, and so forth are of the form:

(sec-constraint-name=*constraint-name*,web-res-list=(*res-list*),user-data-constraint=(desc=*desc*, transport-guarantee=*transport*),auth-constraint=(description=*desc*,name=(*role-list*)))

Where:

Variable	Specifies
constraint-name	The constraint name, which must be unique.
res-list	A comma-separated list of Web resource collection names, defined in the com.sybase.jaguar.webapplication.web-resource-collection property.
desc	An optional text description. Neither of the two description values are displayed in EAServer Manager.
transport	The required network transport security. One of: • NONE – No security required. • CONFIDENTIAL – SSL with data encryption as well as data integrity. • INTEGRAL – SSL with data integrity.
role-list	A comma-separated list of logical role names defined in the com.sybase.jaguar.webapplication.security-roles property.

Usage

For more information on configuring Web application security, see Chapter 3, "Using Web Application Security," in the *EAServer Security Administration and Programming Guide*.

See also

com.sybase.jaguar.webapplication.login-config,
com.sybase.jaguar.webapplication.security-roles,
com.sybase.jaguar.webapplication.web-resource-collection

com.sybase.jaguar.webapplication.security-role.<j2ee-role>

Description

Specifies a mapping from a J2EE role name used in the Web application to a role defined in the EAServer repository.

Syntax

com.sybase.jaguar.webapplication.security-role.*j2ee-role=jag-role*

Where:

- *j2ee-role* is the role name used in the Web application com.sybase.jaguar.webapplication.security-constraint property and listed in the com.sybase.jaguar.webapplication.security-roles property.
- *jag-role* is the role name defined in the EAServer repository.

Usage	In EAServer Manager, set this property using the Role Mapping tab in the Web Application Properties dialog box. Role names may also be specified at the Application or Web application level.
See also	com.sybase.jaguar.webapplication.security-constraint, com.sybase.jaguar.webapplication.security-roles, com.sybase.jaguar.application.security-role.<j2ee-role>, com.sybase.jaguar.application.security-roles

com.sybase.jaguar.webapplication.security-roles

Description	Specifies logical J2EE role names used in the Web application com.sybase.jaguar.webapplication.security-constraint property.
Syntax	*role1*, *role2*, ... Where *role1*, *role2*, and so forth are of the form: (description=*role-desc*, name=*role-name*) Where *role-desc* is an optional description of the role, and *role-name* is the name used in the application.
See also	com.sybase.jaguar.webapplication.security-role.<j2ee-role>, com.sybase.jaguar.webapplication.security-constraint, com.sybase.jaguar.application.security-roles

com.sybase.jaguar.webapplication.servlet-mapping

Description	Associates installed servlets with request paths.
Syntax	Same as the Web application property com.sybase.jaguar.server.servlet.servlet-mapping.
See also	Servlet properties

com.sybase.jaguar.webapplication.session-config

Description	Configures HTTP session properties.
Syntax	(session-timeout=*timeout*) Where *timeout* is the session timeout in minutes. A value of 0 indicates that sessions do not expire.

See also com.sybase.jaguar.webapplication.cookie.persistent

com.sybase.jaguar.webapplication.sessionid

Description Specifies how a Web application sends and receives the HTTP session identifier. The identifier can be sent as a cookie, or it can be encoded in the URL. To encode the session identifier as part of the URL (also known as URL rewriting), servlet and JSP developers must call the HttpServletResponse.encodeURL(String) method or equivalent methods.

Syntax

Table B-24: Session ID storage mechanisms

Value	To indicate
`url`	Session IDs are always encoded in URLs, provided servlet and JSP developers call the encodeURL(String) method or equivalent methods. Cookies are never used to store the session ID.
`cookie`	The generated session ID is returned as a cookie. If the user has disabled cookies, no HTTP session can be created. If encodeURL(String) or other equivalent methods are called, the session ID is not encoded into the URL.
`url,cookie`	When a session is created, the session ID is encoded in the response URL (provided servlet and JSP developers call the encodeURL(String) method or equivalent methods). The ID is also returned in a cookie. When reading an HTTP request from a client, the server tries to obtain the identifier from the current URL; failing that, the session ID is obtained from the cookie if present. If the ID cannot be obtained from the URL or a cookie, the session is not restored.
`cookie,url` (default)	When a session is created, EAServer returns the session ID to the client in a cookie if possible, otherwise the session ID is encoded in the URL (provided servlet and JSP developers call the encodeURL(String) method or equivalent methods). If cookies are supported, the session ID is not embedded in the URL even if encodeURL is called. When reading an HTTP request from a client, the server will try to obtain the identifier from the cookie; failing that, the session ID is obtained from the URL. If no cookie is present, and the URL does not contain a session ID, no session is restored.

Usage

This property applies to both regular and authenticated sessions. You can set this property to specify exactly what session mechanism the client must support when connecting to the Web application.

When URL encoding is used with HTTP connections, it is possible for users to impersonate another user by capturing the encoded URL and resubmitting it on their own connection. To prevent such attacks completely, set this property to `cookie` or use an HTTPS connection with encryption enabled. To minimize the likelihood of such attacks, set the property to `cookie,url`.

com.sybase.jaguar.webapplication.sharecompiledjspclasses

Description

Enables sharing of JSP class files by servers that run from the same EAServer installation.

Syntax

`true` or `false`. The default is `false`.

See also

"Source and class file locations" in Chapter 24, "Creating JavaServer Pages," in the *EAServer Programmer's Guide*.

com.sybase.jaguar.webapplication.small-icon

Description

Specifies the name of the small icon file associated with the Web application. This property is not used in EAServer, but accommodated to comply with the Servlet 2.3 Web archive descriptor.

Syntax

A file name.

com.sybase.jaguar.webapplication.taglib

Description

Configures path aliases for JSP Tag Library Descriptors (TLDs) used in the Web application.

Where *alias* is the path used in JSP source code, and *real-path* is the TLD files location relative to the Web application's context root.

Syntax

alias1,*alias2*,...

Where *alias1*, *alias2*, and so forth are alias definitions of the form:

(taglib-uri=*alias*, taglib-location=*real-path*)

Where:

Variable	Specifies
alias	Is the path used in JSP source code
real-path	Is the TLD file's location relative to the Web application's context root

com.sybase.jaguar.webapplication.web-resource-collection

Description

Specifies a collection of request paths to be protected by security constraints.

Syntax

set1,*set2*,...

Where *set1*, *set2*, and so forth are of the form:

(web-res-name=*name*,http-method=(*methods*),url-pattern=(*patterns*))

Where:

Variable	Specifies
name	The name used to refer to this collection in com.sybase.jaguar.webapplication.security-constraint property. For compatibility with EAServer Manager, use the format: *constraint/display* Where *constraint* is the security constraint that uses this collection, and *display* is the collection name displayed when this resource collection is viewed and edited in EAServer Manager.
methods	A comma-separated list of HTTP methods that can include: `POST`, `GET`, `TRACE`, `DELETE`, `PUT`, `OPTIONS`
patterns	A comma-separated list of URL patterns associated with the collection, for example: /control/*,welcome.htm

See also

com.sybase.jaguar.webapplication.security-constraint

com.sybase.jaguar.webapplication.welcome-file-list

Description

Configures welcome files for the Web application. Welcome files are used to satisfy HTTP requests that end in a directory name, rather than specifying the full path to a file or a path that is mapped to a servlet invocation.

Syntax

A comma-separated list of file names which cannot contain path separators.

com.sybase.jaguar.webapplication.XSLTfactory

Description

Specifies the class name for a custom XSLT XML parser factory class.

Syntax

The class name. Either the class name, or, if the class is in a JAR file, the JAR file name, must be listed in the com.sybase.jaguar.webapplication.java.classes property. The file must be placed in one of the following directories; EAServer searches the directories in the given order:

1 The *WEB-INF/lib* directory under the Web application's context root

2 *$JAGUAR/java/classes*

3 *$JAGUAR/java/lib*

See also

com.sybase.jaguar.webapplication.jagmgr.XSLTFactoryChoice, com.sybase.jaguar.webapplication.DOMfactory, com.sybase.jaguar.webapplication.SAXfactory

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X