SYBASE®

System Administration Guide

EAServer

6.0

DOCUMENT ID: DC38034-01-0600-01

LAST REVISED: July 2006

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 from the Management Console, and changing the administration password.

Chapter 2, "Management Console Overview," describes the Management Console, which is the Web-based graphical user interface to EAServer.

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 data sources, 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, "Exporting Server Modules," explains how to export server entities, back up a server, or synchronize servers and clusters.

Chapter 8, "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 9, “Importing Application Components,” explains how to deploy packages and components, Web applications, J2EE applications, J2EE connectors, and application clients.

Chapter 10, “Configuring Java Class Loaders,” describes how to use custom class loaders to deploy and update components and applications.

Chapter 11, “Runtime Monitoring,” describes how to track EAServer performance and statistics.

Chapter 12, “Command Line Tools,” describes the command line tools that are available in EAServer 6.0.

Chapter 13, “JNI Compiler,” describes the JNI compiler command line tool that can generate C++ proxies for accessing Java classes and interfaces, and C++ classes for implementing Java native methods.

Chapter 14, “Systems Management,” explains how you can monitor and manage EAServer applications, components, and the server itself.

Related documents

Core EAServer documentation The core EAServer documents are available in HTML and PDF format in your EAServer software installation and on the SyBooks™ CD.

What’s New in EAServer 6.0 summarizes new functionality in this version.

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

The *EAServer Automated Configuration Guide* explains how to use Ant-based configuration scripts to:

- Define and configure entities, such as EJB modules, Web applications, data sources, and servers
- Perform administrative and deployment tasks

The *EAServer CORBA Components Guide* explains how to:

- Create, deploy, and configure CORBA and PowerBuilder™ components and component-based applications
- Use the industry-standard CORBA and Java APIs supported by EAServer

The *EAServer Enterprise JavaBeans User’s Guide* describes how to:

- Configure and deploy EJB modules
- Develop EJB clients, and create and configure EJB providers
- Create and configure applications clients

- Run the EJB tutorial

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 Java Message Service User's Guide* describes how to create Java Message Service (JMS) clients and components to send, publish, and receive JMS messages.

The *EAServer Migration Guide* contains information about migrating EAServer 5.*x* resources and entities to an EAServer 6.0 installation.

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

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

- Start the preconfigured server and manage it with the Sybase Management Console
- Create, configure, and start new application servers
- Define database types and data sources
- 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

The *EAServer Web Application Programming Guide* explains how to create, deploy, and configure Web applications, Java servlets, and JavaServer Pages.

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 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://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.eas_5.2.eastg/html/eastg/title.htm.

jConnect for JDBC documents EAServer includes the jConnect™ for JDBC™ 6.0.5 driver to allow JDBC access to Sybase database servers and gateways. The *jConnect for JDBC 6.0.5 Programmer's Reference* is available on the Sybase Product Manuals Web site at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.jconnjdbc_6.05.prjdbc/html/prjdbc/title.htm&toc=/com.sybase.help.jconnjdbc_6.05/toc.xml.

Sybase Software Asset Management User's Guide EAServer includes the Sybase Software Asset Management license manager for managing and tracking your Sybase software license deployments. The *Sybase Software Asset Management User's Guide* is available on the Getting Started CD and in the EAServer 6.0 collection on the Sybase Product Manuals Web site at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.eas_6.0/title.htm.

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 Ant or jagtool to configure applications rather than the Management Console

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` (with *Server* in italic) • 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 the Management Console, 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://sybooks.sybase.com/nav/base.do.

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 Eclipse help formats, which you can navigate using a screen reader.

The Management Console supports working without a mouse. For more information, see "Keyboard navigation" on page 17.

The Web Services Toolkit 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 see the 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 may 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.

CHAPTER 1 Getting Started

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

Starting the preconfigured server

Before you run the preconfigured server for the first time, you must set the password for the primary administrative user. When you install EAServer, the installer prompts you to enter the password. After you install EAServer, you can reset the administrative password using the set-admin-password script.

❖ Setting the password for the administrative user

The account name of the primary administrative user is admin@system. For backward compatibility, jagadmin@system is an alias for admin@system.

- At a command prompt, change to the EAServer *bin* directory.

  On UNIX, run:

  ```
  set-admin-password.sh
  ```

  On Windows, run:

  ```
  set-admin-password.bat
  ```

The system prompts you to enter a password, which must contain at least six characters, and one of those characters must be a digit. To configure these password requirements, see "Configuring domains" in Chapter 10, "Security Configuration Tasks," in the *EAServer System Administration Guide*.

Classes compiled with JDK 1.5 Classes compiled with JDK 1.5 cannot be run in JDK 1.4 or earlier JDK versions. See "Switching back to JDK 1.4 from JDK 1.5" in "Command line syntax" on page 48.

❖ **Starting the preconfigured server in UNIX**

1. By default, EAServer runs with JDK version 1.4, and uses the JVM defined by the java command. "Command line syntax" on page 48 describes how to set these and other options when starting the server.
2. Change to the *$DJC_HOME/bin* directory and run:

    ```
    start-server.sh
    ```

❖ **Starting the preconfigured server in Windows**

- If the preconfigured server is not installed as a Windows service, select Start | Programs | Sybase | EAServer 6.0.0, and choose either:
    - Start Server (JDK 1.4) – starts the server using JDK 1.4 and the Java HotSpot Server VM.
    - Start Server (JDK 1.5) – starts the server using JDK 1.5 and the Java HotSpot Server VM.

To start a user-defined server, you must first create the server. For instructions, see "Creating or deleting a server" on page 21.

Using the Management Console

You can use the Management Console to configure EAServer and to define and deploy software components. To use the Management Console to manage EAServer, you must be the admin@system user or belong to the admin-role role.

For additional information on EAServer administrative privileges, see Chapter 10, “Security Configuration Tasks,” in the *EAServer System Administration Guide*.

Connecting to EAServer

To connect to EAServer, open a Web browser and enter the following, replacing <*host*> with the name of the machine where EAServer is running:

```
http://<host>:8000/console
```

8000 is the HTTP port number for the preconfigured server. To connect to another server, enter its HTTP port number instead of 8000.

EAServer logs errors and other messages in *<server_name>.log*, in the *logs* subdirectory.

❖ Connecting to a server

- In the Login window, enter `admin` as the user name, and enter the password that you set for this user.

  The Management Console connects to the preconfigured server.

Disconnecting from EAServer

The Management Console allows you to disconnect from a server so that you can connect to another server, or reconnect to the same server, without restarting the Management Console.

❖ Disconnecting from a server

1. Highlight the server, right-click, and select Disconnect.
2. In the wizard, confirm that you want to disconnect.

Administration password and OS authentication

Members of the admin-role role have unlimited access to EAServer through the Management Console. Initially, the admin@system user is the only member of this role. For additional security, you can enable operating system authentication.

Administration You must establish an administrative password for the administrative user on each server—see "Setting the password for the administrative user" on page 1. The administrative user can:

- Access EAServer through the Management Console
- Set or reset the admin@system password
- Enable and disable user authentication

Enabling OS authentication If defined, OS authentication 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

- Configure an effective user name and group for the server to run as—see "Security tab" on page 37.

❖ 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. Set the login method for each security domain that will use OS authentication:
   - a In the Management Console, expand the Security icon, then expand Domains, and select the security domain.
   - b On the General tab, select "os-auth" for the Login Method. Select Apply.

❖ 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, select Performance and Maintenance.
3. Select 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 Set the login method for each security domain that will use OS authentication:

   a In the Management Console, expand the Security icon, then expand Domains, and select the security domain.

   b On the General tab, specify “os-auth” as the Login Method. Click Apply.

JAAS To use Java Authentication and Authorization Service (JAAS), see Chapter 8, “Using the JAAS API,” in the *EAServer Security Administration and Programming Guide*.

Shutting down a server

To use the Management Console to shut down a server, the Management Console must be connected to the server.

1 In the Management Console, expand Servers, then select the name of the server you want to shut down.

2 Right-click, and select Shut Down Server.

3 Confirm that you want to stop the server.

For command line options, see “Stopping servers” on page 51.

Verifying your environment

If you have any problems running EAServer:

All platforms

- Check the server log file (*<server_name>.log*) in the *logs* subdirectory for error messages.
- Verify the server’s HTTP and IIOP port settings. Default listeners are defined for each protocol, but you may have changed them. See “Configuring listeners” on page 41 for more information.
    - The HTTP port is used by EAServer to listen for HTTP requests. The default is 8000.

    - IIOP is the port used by EAServer to accept IIOP requests. The default is 2000.
- Verify the values for variables that are set in *bin\djc-setenv.bat* or *djc-setenv.sh*.

Warning! Do not edit the djc-setenv script. If changes are required, create a *local-setenv.bat* or *local-setenv.sh* file in the *bin* directory, and set the correct values. djc-setenv calls local-setenv when you start the server.

UNIX

Verify that the THREADS_FLAG environment variable is either not set or is set to a value that matches the setting in the *djc-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://infocenter.sybase.com/help/index.jsp. In the left pane, select EAServer 5.2 Core Documentation, then select the *EAServer Troubleshooting Guide*.

CHAPTER 2 Management Console Overview

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

Topic	Page
Overview	9
Using the Management Console	10
Keyboard navigation	17

Overview

In EAServer 6.0, the Sybase Management Console, which is a Web-based user interface, provides the graphical user interface to EAServer. The Management Console offers functionality similar to Sybase Central, including support for servlet and JSP-based Web applications. The Management Console uses JavaScript, but also functions correctly if JavaScript is disabled.

The Management Console provides graphical administration facilities for EAServer, including support for deployment and runtime monitoring of applications. The Management Console enables users to perform administrative tasks from a Web browser, including managing files, monitoring system components, and editing text files.

Throughout the rest of this book, the Sybase Management Console is called the Management Console.

❖ Accessing EAServer using the Management Console

To use the Management Console, you must be the admin@system user or belong to the admin-role role. If you are connecting for the first time, set the password for the admin@system user as described in "Setting the password for the administrative user" on page 1.

To manage EAServer, log in to the Management Console, then connect to the server.

1 In a Web browser, access the following URL, where *host* is the name of the machine where EAServer is running and *port* is the HTTP port number. The default port number is 8000.

```
http://host:port/console/Login.jsp
```

2 In the login window, enter your user name and password, and click Login.

The Management Console connects to the preconfigured server.

Using the Management Console

Using the Management Console, you can manage EAServer and its components, configure the plug-ins, and customize display properties. This section describes:

- Managing EAServer
- Using wizards
- Running tutorials
- Refreshing the Management Console display
- Configuring the EAServer Manager plug-in
- Configuring the Sybase Web Services Toolkit plug-in
- Configuring the Web Services Registries plug-in
- Setting Management Console display preferences
- Logging out of the Management Console

Right-click menus require JavaScript You can perform actions in the Management Console using either the menus that display when you select and right-click a node or the Actions menu at the bottom of the Console. To display the right-click menus, JavaScript must be enabled in your browser. This manual describes how to use the right-click menus.

Managing EAServer

Use the Management Console to configure EAServer and to define and deploy software components.

Application developers can use the Management Console to view the Ant configuration files that define EAServer and its components, and to create and edit user-configuration files for installed modules.

To simplify application deployment, the Management Console defines the following basic, middle-tier application units:

- Application – a group of packages and Web applications bundled into a single unit.
- Cluster – a set of servers that share configuration information and run the same set of components.
- Component – contains the methods that execute business logic and access data sources.
- Export configuration – a set of files, Java classes, and entities (application clients, applications, connectors, EJBs, and Web applications) bundled into a single unit for easy deployment between servers.
- Server – an EAServer runtime process with its own network addresses for client session connections and for HTTP (HTML) connections.
- 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 and servers, which lets you test and debug component implementation changes without restarting the server.

Logging errors

When you run the Management Console, logging and error messages are written to the *<server_name>.log* file, in the EAServer *logs* directory.

The Management Console enables you to view server log files remotely and to monitor statistics for component execution and network activity. For more information, see Chapter 11, “Runtime Monitoring.”

Using connection profiles

When you log in to the Management Console, you can use a predefined connection profile, which speeds up the connection process. Define profiles using the New Connection Profile wizard. See “Creating connection profiles” on page 22.

Using wizards

EAServer provides wizards for creating, deploying, and configuring entities such as applications, clusters, EJB modules, export configurations, Web applications, and so on. Each wizard contains the following pages:

- **Introductory page** The introductory page describes the entity that you are creating or deploying. You can select to skip this page the next time you run the wizard.
- **Configuration pages** One or more pages in which you configure property values for the entity.
- **Summary page** Displays a summary of the values on the configuration pages.

Each page contains the following buttons:

Button	Click to
Next	Continue to the following page.
Back	Return to the previous page.
Finish	Complete the wizard and accept default values for the remaining properties. The Finish button is disabled until you configure the required properties.
Cancel	Exit the wizard, and cancel the procedure.
Help	Display online help.

Where appropriate, buttons are disabled; for example, on the Introductory page, the Back button is disabled.

Running tutorials

The Management Console tutorials demonstrate how to:

- Configure and use the Management Console – access the tutorial on the main Management Console window, by clicking Sybase Management Console Tutorial.
- Connect to EAServer, deploy J2EE applications, set up logging, view statistics, and configure a data source – access the tutorial by selecting the EAServer Manager plug-in, then clicking the link for the tutorial.

Refreshing the Management Console display

To refresh the Management Console view, select one of the following options from the context menu:

- **Refresh** Queries the server to update the properties for items that are linked to this level of the tree view, and updates the display to reflect changes, additions, or deletions.
- **Refresh Node** Reloads the tree view display returned from the server. This is the same as refreshing the page with your Web browser's menu or hot key.
- **Refresh Domain** Applies only to active connections, which display directly beneath EAServer Manager in the tree view. Refreshes the properties for all entities displayed beneath this node.

Configuring the Management Console plug-ins

You can configure general properties for the Management Console plug-ins.

❖ Configuring the EAServer Manager plug-in

1. In the left pane, expand the Preferences folder, then expand Plug-Ins, and select EAServer Manager.
2. On the General tab, select:
   - The domain to configure.
   - Whether to synchronize properties to their corresponding XML configuration file. If selected, the values in the Ant configuration script that defines an entity are synchronized with the values set in the Management Console.
   - Whether to display EAServer system components.
3. Click Apply to save your changes.

❖ Configuring the Sybase Web Services Toolkit plug-in

1. In the left pane, expand the Preferences folder, then expand Plug-Ins, and select Sybase Web Services Toolkit.
2. On the General tab, select the domain and the server to configure, then enter:
   - **Machine Name** The name of the machine on which EAServer is running.

- **Protocol** Select HTTP or HTTPS for Web service connections.
- **HTTP Port** The port number to use for HTTP or HTTPS connections.
- **User ID** A user name for connecting to the server.
- **Password** A valid password for the user name.
- **Auto Connect on Console Login** Select to connect automatically, using the options defined in this window, when you log in to the Management Console.

3 Click Apply to save your changes.

For information about how to create and manage Web services in EAServer, see the *EAServer Web Services Toolkit User's Guide*.

❖ Configuring the Web Services Registries plug-in

1 In the left pane, expand the Preferences folder, then expand Plug-Ins, and select Web Services Registries.

2 On the General tab, select the registry to configure, then enter:

- **Registry Profile Name** The Web service name that displays in the registry.
- **Query URL** The URL to use to get information from the Web service registry.
- **Publish URL** The URL for publishing information to the Web service registry.
- **User Name** The login name for a user that has access to the Web service registry.
- **Password** A valid password for the user.
- **Auto Connect to Registry Server** Select to automatically log in to the Web services registry when you log in to the Management Console.

3 Click Apply to save your changes.

For information about Web services registries, see the *Web Services Toolkit User's Guide*.

Customizing the Management Console display

To customize how the Management Console displays some graphical elements, set display preferences.

❖ **Setting Management Console display preferences**

If you update any property values, click Apply to save your changes.

- In the left pane, highlight Preferences, and configure display properties in the right pane, on these tabs:
  - General tab
  - Behavior tab
  - Advanced tab

General tab

On the Preferences General tab, enter:

- **Location of the Apply, Reset, and Cancel Buttons** Specify where to display the buttons:
  - Top Left
  - Top Right
  - Bottom Left (the default)
  - Bottom Right
  - Far Bottom Left
  - Far Bottom Right
- **Show Property Descriptions** Select to display property descriptions on the property sheets.
- **Character Encoding** Select the character encoding method to use for converting bytes into characters:
  - ISO-8859-1 (English)
  - GB2312 (Chinese)
  - Shift_JIS (Japanese)
- **Color Scheme** Select the color scheme:
  - Blue (the default)

- Sybase Central
- Beige

Note To effect a color change, clear the browser's cache and refresh the page.

Behavior tab

On the Preferences Behavior tab, enter:

- **Show Wizards In** Specify how to display wizards:
  - Right Pane (the default)
  - Modal Pop-Up Window (requires JavaScript)
  - New Browser Window (requires JavaScript)
- **Show Category Nodes In** Specify where to display the category nodes:
  - Tree View Only (the default)
  - Tree View and Property Sheets
- **Confirm Deletions** If selected, you must confirm requests to delete an entity.
- **Enable Double-Click Support** Select to enable expanding a node in the tree view by double-clicking the item, instead of selecting the plus sign.
- **Default Node Selection on Login** Select what to display when you log in to the Management Console:
  - None – displays the Welcome window (the default).
  - Select the Node That Was Selected Before Your Previous Logout – displays your previous selection.
  - Select the Node Named – displays the node you select from the list.

Advanced tab

On the Preferences Advanced tab, specify:

- **Maximum HTTP Form Content** The maximum number of bytes allowed in an HTTP form. If you update this value, select Apply, then restart the server for your change to take effect.

Logging out of the Management Console

When you finish working with the Management Console, before you navigate to another location or close the browser, log out of the Management Console by clicking Logout in the upper-right part of the window.

Keyboard navigation

The Management Console allows you to work without a pointing device. You can navigate with the keyboard as follows:

- **Moving 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 Management Console icon.

  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 display for the item you have highlighted in the tree view.

  d Details view data, the right window pane where the Management Console 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.

- **Navigating in the tree view and details view** The tree view, in the left pane of the Management Console, 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 views, first move the focus to the tree view or Detail view, then use these keys:

  - Enter – collapse or expand items in the tree view.

- Tab – in the details view, use the Tab key to move the focus to the desired control. To activate buttons, place the focus on the button and press Enter. You can display online help for the displayed tab by placing the focus on the Help button and pressing Enter.

  Note If you press the Enter key on a wizard page or properties page in the Management Console and the focus is not on a button, the button that is activated is browser-specific. Therefore, verify that the focus is on the correct button before pressing the Enter key.

- **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 through the toolbar buttons, table-of-contents pane, and search pane	Ctrl+Tab to move the focus forward and Ctrl+Shift+Tab to move backward. On most systems, toolbar buttons indicate that they have the focus by showing a dotted line around the button label. 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 on top of 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 expand topics with a + sign 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, then press Tab. Type the text to search for, then press Enter. When the results appear, place the focus in the table-of-contents pane, then press Tab once to move the focus to the list of results. Use the arrow keys, along with the Page Down, Page Up, Home, and End keys to scroll through the results. Press Enter to display the text of the selected item.
Place the focus in the text (right) pane	Ctrl+Tab to move the focus to the right pane, then press Tab.

To do this	Use these keys
Navigate in the text pane	Place the focus in the text pane as described above. To scroll, use the arrow keys, along with the Page Up, Page Down, Home, and End keys. To move the focus to hyperlinks, use Tab to move the focus to the next hyperlink, and Shift+Tab to move the focus to the previous hyperlink. To follow the hyperlink that has the focus, press Enter.
Activate a toolbar button (Back, Forward, Print, Page Setup)	Place the focus on the button and press Enter.
Scroll the table-of-contents	Place the focus in the table-of-contents pane. Press F8 to place the focus on the splitter bar. Use the left-arrow and right-arrow keys to scroll the table-of-contents.

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 Management Console.

Creating or deleting a server

❖ **Creating a server**

1 In the Management Console, highlight the Servers folder, right-click, and select Add.

2 Complete the wizard to create the server:

- If the name of the server is different than the name of the machine on which it runs, enter the fully-qualified machine name as the server host name. For example, if the name of the server is "myServer" and it runs on a machine called "mach1" in the Sybase domain, enter `mach1.sybase.com` as the server host name.
- If you are adding a server to a cluster and running it on the same machine as other servers in the cluster, create new HTTP and IIOP listeners for the server. If all the servers in a cluster run on separate machines, all the servers can use the same listener port numbers.

3 To connect to the server, create a new connection profile and specify the new server's IIOP port number—see "Creating a connection profile" on page 22.

4 Start the server. See "Starting the server" on page 47.

The server logs errors and other messages to the *logs\<server_name>.log* file, where *server_name* is the name of the server.

❖ **Deleting an existing server**

1 Expand the Servers folder, highlight the server to delete, right-click, and select Delete.

2 Complete the wizard, and select Finish.

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

Creating connection profiles

When you log in to the Management Console, you can use the predefined connection profile "Local Server," or you can define a new connection profile. If you create a new server, create a new connection profile to use for connecting to the new server.

❖ **Creating a connection profile**

1 Highlight EAServer Manager, right-click, and select Create Connection Profile.

2 Define the following fields:

- **Profile Name** The name of the profile that displays in the Management Console.
- **Host Name** The name of the host to which you are connecting.
- **Protocol** Select the protocol, IIOP or IIOPS.
- **Port Number** The port number on the host to which you are connecting.
- **User ID** The ID of the user connecting to the host; for example, admin@system.
- **Password** A valid password for the user.
- **Auto Connect to Server on Console Login** Select whether to connect automatically using this profile when you log in to the Management Console.

3 After the wizard completes, the right pane displays the property values for this profile. If you change any values, click Apply to save your changes.

❖ Connecting to a server using a connection profile

- Highlight the connection profile, right-click, and select Connect. The default profile name is “Local Server.”

❖ Deleting a connection profile

1 If an active connection is using the profile you want to delete, highlight the profile, right-click, and select Disconnect.

2 To delete the profile, highlight the profile, right-click, and select Delete.

3 Complete the wizard, and select Finish.

Configuring servers

In the Management Console, property tabs allow you to configure and tune a server.

❖ Configuring or modifying server properties

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

1 From the Management Console, display the list of installed servers by expanding the Servers folder.

2 Highlight the server to configure.

3 The server properties display in the right pane on these:

- General tab – define general server properties.
- HTTP tab – determine browser accessibility.
- Listeners tab – enable or disable listeners.
- Log/Trace tab – set logging and trace options.
- Modules tab – select modules to start automatically.
- Monitors tab – set performance and memory monitors to tune server performance.
- Performance tab – limit ODBC calls to a single thread.
- Resources tab – specify resources to be activated when the server starts.
- Security tab – configure secure connections and specify UNIX user and group name.
- Services tab – select service components to start automatically.
- Tasks tab – schedule tasks to be performed automatically.

If you modify server properties, click Apply to save your changes, or click Reset to discard the changes. You must restart the server for the changes to take effect. To restart the server, highlight the server icon, right-click, and select Restart.

Note The Property column for each tab lists the property names as they display in the Management Console. The one-word property names written in "lowerCamelCase" (for example, processID) are the corresponding configuration property names.

General tab

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

Table 3-1: Server general properties

Property	Description
Statistics URL	Select to display a spreadsheet that contains server statistics.

Property	Description
Host Name	The name of the machine on which the server is running. The repository property name is host.name.
Process ID	A unique ID for the server instance, in the form of a dotted-decimal TCP/IP address, a colon, and a port number; for example, 10.11.12.13:2000. The processID is used to construct session IDs, so it must be unique across the network of connected servers that share the same databases.The ID need not contain an actual TCP/IP address; the default value is derived from either the iiopListeners or the httpListeners property, but you can redefine it.
Java Class Path	Classes in this path are loaded before classes in the server class path at server start-up. **Note** Sybase recommends that you define this path carefully; incorrect use may interfere with server operation. Alternately, you can add extension JAR files, which are loaded after classes in the server class path, to the EAServer *lib/ext* directory. If classes are required by user-spawned threads, add them either to the Java class path or to the *lib/ext* directory. For loading classes used by EJB modules and Web applications, you can configure named class loaders at the server, application, or package level—see Chapter 10, “Configuring Java Class Loaders.” The javaClassPath property has no default value.
Java Start-up Options	java command line options used to start the server. If you are using a Sun JVM, use the -server option to activate the HotSpot Server VM. The javaStartupOptions property has no default value.
License Configuration	Select the server license configuration from the list. Initially, featureManagerConfig is set to “default.” See “Configuring the server license” on page 54.
Default Code Set	Select the character encoding to use; the default value of defaultCodeSet is utf8 (8-bit Unicode Transformation Format).

Property	Description
Allow Runtime Compile	Select to allow runtime compilation of Java source code. If the code has changed since the last time it was compiled, the server recompiles the code for IIOP interface stubs, SQL persistence classes, and JavaServer Pages (JSPs). The default value of allowRuntimeCompile is true. IIOP interface stubs and SQL persistence classes are precompiled automatically when you deploy EJB modules. If your server is running under a Java Runtime Environment (JRE) without a Java compiler, unselect this property. If you change the configuration of a data source that is used by EJB entity beans, and this property is unselected, you must redeploy the EJB-JAR files and restart the server. If you are using JSPs and this property is unselected, your JSPs must be precompiled. If you define all JSPs that need to be precompiled with their own “servlet” elements in your *web.xml* files, you can precompile them using the deploy tool. See deploy on page 174. The deploy tool requires the Java compiler, so if the target machine does not have a Java compiler, you can run deploy on one machine and transfer the deployed configuration to the target machine. To transfer deployed configurations between machines, you can use an export configuration—see Chapter 7, “Exporting Server Modules.”
Disable Statistics	Select to disable all server statistics. The default value of disableStatistics is false. Selecting this option improves performance, but provides much less information about server actions. You may want to compare the difference in performance to determine whether the lack of statistics is worth the performance gain.
Disable 60-Second Statistics	Select to disable 60-second summaries for all available statistics; disableStatistics must be true. In addition to the statistical information described in Chapter 11, “Runtime Monitoring,” 60-second summary information includes: • For count statistics, the change in the last 60 seconds. • For range statistics, the minimum and maximum values for the last 60 seconds. • For time statistics, the changes in total time, and minimum and maximum time values for the last 60 seconds. The default value of disable60SecondStatistics is false. 60-second summary information is updated once per minute, for the previous full minute.
Disable 60-Minute Statistics	Select to disable 60-minute averages for all available statistics; disableStatistics must be true. In addition to the statistical information described in Chapter 11, “Runtime Monitoring,” 60-minute summary information includes: • For count statistics, the change in the last 60 minutes. • For range statistics, the minimum and maximum values for the last 60 minutes. • For time statistics, the changes in total time, and minimum and maximum time values for the last 60 minutes. The default value of disable60MinuteStatistics is false. 60-minute summary information is updated once per hour, for the previous full hour.

Property	Description
Enable Reverse Host Lookup	If enabled, the server makes network calls to request the host names of client machines, to use for statistics. Some customers do not support these queries on their network. If disabled, performance improves because EAServer does not attempt to get client host names. The default value of reverseHostLookupEnabled is true.
Enable XML Entity Cache	Select to cache XML entities used by the deploy tool. XML entities (such as external DTDs) are cached in the *XmlEntityCache.properties* file, in the *Repository/Component/com/sybase/djc/xml* directory. You can delete this file to clear the cache. The default value of enableXmlEntityCache is true.
Enable XML Grammar Cache	Select to cache XML grammars used by the Xerces parser. The default value of enableXmlGrammarCache is true.
Work Offline	Select to disable JMS store and forward operations at server start-up time. You can activate JMS store and forward operations dynamically (once the server is running) in Java code by setting the Java system property jms.workOffline to true. The ability to disable JMS store and forward operations is most useful on mobile devices, where running applications may need to control whether background JMS operations use the network. By default, the workOffline property is false.

HTTP tab

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 3-2.

Table 3-2: HTTP properties

Property	Description
Verify IP for Session	If the IP address where the session was created does not match the IP address of the request, specify what action to perform: invalidate, log, or none. The default value of verifyIPForSession is invalidate.
HTTP Contexts	To enable the defined HTTP contexts, select: • All – enables all HTTP contexts. • None – disables all HTTP contexts. • Select – enable or disable HTTP contexts individually. If you click the link for a defined HTTP context, its configuration tab displays. Initially, the httpContexts property is set to the default HTTP context. See "Adding and configuring HTTP contexts" on page 29.

Property	Description
Domain Name	The Web server redirector's domain name. Proxy support requires information about the machine hosting the Web redirector. Because the Web redirector may be in a domain, other than the one EAServer is in, the server property httpDomain may be required to ensure that the redirector's domain is used, instead of the server's domain.
Proxy Protocol	The default protocol used to connect to the Web redirector, HTTP or HTTPS. The name of the configuration property is httpProxyProtocol.
Proxy Port	The HTTP or HTTPS port of the Web redirector. The protocol is defined by httpProxyProtocol. The value of httpProxyPort is used when the server must redirect requests automatically, such as for form-based authentication and client-side redirects.
Get Server Info From	Select one of proxy, server, or source. The configuration property name is httpGetServerInfoFrom. If you call the HttpRequest methods, getServerName, getServerPort, or getProtocol, the value of httpGetServerInfoFrom determines how the server name, port number, and protocol are retrieved—see Table 3-3 on page 28. The default value is proxy. However, if the required properties—httpDomain, httpProxyPort, and httpProxyProtocol—are not set, source is used, instead of proxy. To create a URL dynamically from a JSP or servlet, you can use HttpRequest object methods to get the server host, port, and protocol. If the server is configured with a Web redirector, you may want JSPs and servlets to redirect requests to the Web redirector, instead of the server.

Table 3-3 defines the return values for HttpRequest methods, which you can call from JSPs or servlets to create URLs dynamically. The return value for each method depends on the value of the server property httpGetServerInfoFrom, which can be one of proxy, server, or source. See Table 3-2 on page 27.

Table 3-3: HttpRequest object methods

Method	Proxy	Server	Source
getServerName	Gets the name of the server from the httpDomain property.	Gets the name of the server to which requests are redirected	Gets the server name that is entered in the browser.
getServerPort	Gets the proxy port from either the httpProxyPort or httpsProxyPort property.	Gets the port number of the server to which requests are redirected.	Gets the server port number that is entered in the browser.
getProtocol	Gets the protocol from the httpProxyProtocol property.	Gets the protocol that was used to connect to the server.	Gets the protocol that is entered in the browser.

Adding and configuring HTTP contexts

An HTTP context defines the context path that is used to access a server's document root, HTML pages that display, and allowed operations.

❖ Adding an HTTP context

1. Select the HTTP Contexts folder, right-click, and click Add.
2. In the wizard, enter a name for the HTTP context, and select Finish.

❖ Configuring an HTTP context

1. Expand the HTTP Contexts folder, and select the HTTP context to configure.
2. In the right pane, edit the following properties, then click Apply:
   - **Context Path** The URL context to access the document root. The configuration property name is contextPath, and the default value is "/".
   - **Resource Base** The document root for HTML files. The configuration property name is resourceBase. The value of resourceBase for the default HTTP context is *${djc.home}/html*.
   - **Welcome Files** The HTML file that displays when users access the document root. If you enter a comma-separated list of files, the first file in the list displays, unless it is not available, in which case the second file in the list displays, and so on. To display welcome files, select Redirect to Welcome Page. The configuration property name is welcomeFiles.
   - **Handlers** A comma-separated list of error handlers. The configuration property name is handlers. For the default HTTP context, the value is "Resource, NotFound."
   - **Allowed Methods** Select the methods that are allowed to be performed in a server's context; the default value is "GET." The configuration property name is allowedMethods.
   - **Directory Browsing Allowed** Select to enable users to browse server directories. The configuration property name is dirAllowed.
   - **Redirect to Welcome Page** Select to display a welcome file when users access the document root. The configuration property name is redirectWelcome. If this property is not selected, the value of welcomeFiles is ignored.

Listeners tab

On this tab, you can enable or disable the listeners that are defined for this server. For HTTP, IIOP, and RMI listeners, select one of:

- **All** To enable all the listeners.
- **None** To disable all the listeners.
- **Select** To enable or disable listeners individually.

If you click the link for a defined listener, its configuration data displays.

To define listeners, see "Adding and configuring listeners" on page 42.

Log/Trace tab

Tracing provides information about activities carried out by an 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, and select or unselect the properties described below.

You can also enable these logging and tracing properties on the command line or in the *setenv.bat[sh]* file—see "Configuring system logging" on page 32.

For information on viewing the log files, see "Viewing server log files" on page 155.

- **Capture Console Output in Log** Writes messages that display in the console to the server log file. The name of the configuration property is captureConsoleOutput.
- **Echo Deploy Log to Console** When you deploy a module to EAServer, displays the deployment status in the console. The name of the configuration property is deployConsoleOutput.
- **Echo Server Log to Console** Displays the contents of the server log file in the console. The name of the configuration property is echoLog.
- **Enable HTTP Request Log** Generates an HTTP request log. Logs are named *serverName-http-YYYY-MM-DD.log* and created in the *logs* subdirectory. *serverName* is replaced with the name of the server and *YYYY-MM-DD* is replaced with the current date. The name of the configuration property is enableHttpRequestLog and the default value is false.

- **Enable java.sql.DriverManager Log** Writes DriverManager log messages to the server log. If not selected, messages are discarded. The name of the configuration property is enableDriverManagerLog.
- **Generate Exception Cross Reference** Select to cross-reference exceptions in the server log file. The name of the configuration property is exRef, and the default value is false.
- **Generate Transaction Cross Reference** Select to cross-reference transactions in the server log file. The name of the configuration property is txRef, and the default value is false.
- **Log Application Exceptions** Write application exceptions to the server log. The name of the configuration property is logApplicationExceptions, and the default value is true.

  If a deployed module requires that application exceptions be logged, enable exception logging by setting either the ejb.logExceptions (EJB) or web.logExceptions (Web applications) property in the module's XML configuration script to true. You can edit this script in the Management Console, using the module's Configuration tab.
- **Log System Exceptions** Write system exceptions to the server log. The name of the configuration property is logSystemExceptions, and the default value is true.

  If a deployed module requires that system exceptions be logged, enable exception logging by setting either the ejb.logExceptions or web.logExceptions property in the module's XML configuration script to true. You can edit this script in the Management Console, using the module's Configuration tab.
- **JMX Logging Level** Select the logging level for the Systems Management JMX agent. The name of the configuration property is mx4jLoggingLevel, and the default value is error. See "Running the SNMP subagent" on page 224.
- **Enable RMI-IIOP Trace** Traces remote method invocations for IIOP and RMI-IIOP in the server log. The name of the configuration property is rmiTrace, and the default value is false. You can also enable RMI-IIOP tracing on the command line, which does not change the value in the server properties file.

Warning! Enabling this feature may result in plain-text passwords appearing in the server log, depending on the authentication mechanism used by remote clients.

- **Enable RMI Local Trace** Traces local RMI method invocations (intercomponent calls) in the server log. The name of the configuration property is rmiTraceLocal, and the default value is false.
- **Enable EJB Trace** Traces EJB component invocations in the server log. The name of the configuration property is ejbTrace, and the default value is false.
- **Enable Web Trace** Traces Web component invocations in the server log. The name of the configuration property is webTrace, and the default value is false.
- **Enable JMS Trace** Traces Java Message Service (JMS) operations in the server log. All public and protected JMS provider methods are traced. The name of the configuration property is jmsTrace, and the default value is false.
- **Enable SQL Trace** Traces JDBC driver activity in the server log, including JDBC prepared statement operations, parameter and result information for container managed persistence operations (queries and updates), and transaction commit/rollback operations. The name of the configuration property is sqlTrace, and the default value is false.
- **Server Log Files** Select the log file to view.

Configuring system logging

Logging by EAServer classes and Java clients is written to a log file whose name is determined by the system property djc.logFile. If you do not explicitly set this property, the default file name is *default.log*, which is created in the server's *logs* subdirectory.

Table 3-4 on page 33 defines logging properties that you can set either on the command line, or in either *local-setenv.bat* (Windows) or *local-setenv.sh* (UNIX).

You can also enable the logging and tracing properties defined on the server's Log/Trace tab, by prepending "djc." to the configuration property name, and setting the value to true. For example, to enable EJB tracing, set `djc.ejbTrace=true`; to echo logging to the console, set `djc.echoLog=true`.

Table 3-4: Logging properties

Property	Description
djc.logFile	The name and path of the log file.
djc.logFileReUse	Specify whether to overwrite the contents of the log file when it reaches its maximum size; true or false.
djc.logFileMaxSize	The maximum file size. Specify one of: • `-1` to indicate no limit • ***N***`m`, where *N* is the number of megabytes • ***N***`k`, where *N* is the number of kilobytes
djc.logFileRotation	Specify whether to create a new log file when the existing log file reaches its maximum size; true or false. If set to true, log files are named sequentially; for example, *mylog.log.001*, *mylog.log.002*, and so on.
djc.logFileArchive	Specify whether to archive the log file; true or false.
djc.logFileArchiveName	To archive the log file, specify the name of the archive file. The value of djc.logFileArchive must be true.
djc.logFileArchiveCompress	Specify whether to compress log file archives; true or false. If set to true, the log file archive is compressed to a file named ${djc.logFileArchiveName}*.zip*

❖ **Setting logging options on the command line**

- To set logging options on the command line, use:

```
-Ddjc.logFile=/myserver/myserver.log -Ddjc.logFileMaxSize=5m
-Ddjc.logFileRotation=true
```

When you run a Java client, you can specify logging options on the command line; for example:

```
%JAVA_HOME%\bin\java -Ddjc.rmiTrace=true
-Ddjc.logFile=%DJC_HOME%\logs\rmiClientTrace.log myClient
```

❖ **Setting logging options in *local-setenv.bat* or *local-setenv.sh***

- To specify log options in *local-setenv.bat* or *local-setenv.sh*, so they are set every time you start the server, use:

```
set DJC_JVM_ARGS=%DJC_JVM_ARGS% -Ddjc.logFile=/myserver/myserver.log
-Ddjc.logFileMaxSize=5m -Ddjc.logFileRotation=true
```

See “Setting environment variables” on page 47.

Modules tab

You can select any deployed module to start automatically when the server starts. If you choose to start an application, all of its EJBs and Web modules are also started, even if they are not explicitly listed.

Table 3-5 describes the system modules that are deployed in a standard EAServer installation. By default, they all start automatically.

Table 3-5: System modules

Module	Description
application-console	Provides the Management Console user interface
ejbjar-management	Provides systems management functionality
webapp-sybasewst	Implements the Web Services Toolkit
webapp-wfs	Provides FTP service
webapp-wlb	Provides HTTP redirects to achieve load balancing for other Web applications
webapp-wsh	Enables running shell commands on remote servers

To specify which system modules to start, under System Start Modules, choose Select, then select or unselect each module. To start all the modules, select All. You can also select None.

You can exclude system modules from starting by selecting All or Select under System Exclude Modules. For example, to start all but one of the modules:

1. Under System Start Modules, select All.
2. Under System Exclude Modules, choose Select, then select the module that you do not want to start.

   The next time the server starts, all the modules are started except the one selected under System Exclude Modules.

The configuration properties systemStartModules and systemExcludeModules define the system modules to start or exclude, as comma-separated lists.

If you deploy a new module, it displays under User Start Modules and User Exclude Modules. You can select user modules to start automatically, the same as you do for system modules. The configuration properties that define which user modules to start or exclude are startModules and excludeModules.

Monitors tab

You can configure monitoring for a server by setting the properties on the Monitors tab. You can also tune listener performance by configuring these properties on the listener's Performance tab.

- **Maximum Response Time** The maximum response time (in milliseconds) before the number of active threads is decreased. To specify an unlimited response time, enter `-1` (the default). The configuration property name is maximumResponseTime.
- **Performance Thread Monitor** The thread monitor used for performance monitoring; the default is "performance." The configuration property name is performanceThreadMonitor. See "Monitoring threads" on page 45.
- **Maximum Virtual Memory** The maximum amount of virtual memory that the server process can use, specified in bytes. The configuration property name is maximumVirtualMemory.
- **Critical Memory Limit (%)** The percentage of memory that must be in use before the condition is considered critical and no further requests are accepted. To specify no limit, enter `-1`; the default is 90. The configuration property name is criticalMemLimit.
- **Memory Limit for Alarm (%)** The percentage of memory that must be in use before a warning is logged; the default, -1, indicates that there is no limit. The configuration property is alarmMemLimit.
- **Critical JVM Memory Limit (%)** The percentage of JVM memory that can be in use before the condition is considered critical and no further requests are accepted; -1, indicates that there is no limit. The default value is 90. The configuration property name is criticalJavaMemLimit.
- **JVM Memory Limit for Alarm** The percentage of JVM memory that can be in use before a warning is logged; -1, indicates that there is no limit. The configuration property name is alarmJavaMemLimit.
- **Alarm Log Interval** The frequency at which alarm conditions are logged, specified in seconds; the default is 30. The configuration property name is alarmLogInterval.
- **Memory Thread Monitor** The thread monitor used for memory monitoring; the default is "memory." The configuration property name is memoryThreadMonitor. See "Monitoring threads" on page 45.

Performance tab

You can improve server performance by enabling the following property on the Performance tab.

- **Single-Threaded ODBC Calls** Limit ODBC calls to a single thread. The configuration property name is singleThreadedOdbc; the default value is false.

For information about improving EAServer application performance, see the *EAServer Performance and Tuning Guide*.

Resources tab

On the Resources tab, you can select resources to be activated when the server starts. Typically, resources are started the first time they are used. You may want to configure a resource for activation at server start-up for reasons that include:

- If you are using JMS store and forward (pull or push), to ensure that messages are propagated between servers, even if they are not currently being consumed at the target server.
- If you are using XA two-phase commit, to ensure that recovery processes run for data sources that are not in use, but were in use when the server was previously running.

Resources include database types, data sources, mail sessions, JMS message queues and topics, and connection factories. The configuration property startResources is a comma-separated list that defines which resources to start.

The following resources are predefined:

Resource type	Resource name
Connection factory	ConnectionFactory
	default
	javax.jms.ConnectionFactory
Data source	cluster.db
	default
	JavaCache
	message.db
	session.db
	tx_manager

Resource type	**Resource name**
Message queue	javax.jms.TemporaryQueue
	testQueue
Message topic	javax.jms.TemporaryTopic
	testTopic
Queue connection factory	default
	javax.jms.QueueConnectionFactory
	QueueConnectionFactory
Topic connection factory	default
	javax.jms.TopicConnectionFactory
	TopicConnectionFactory

To activate resources individually, choose Select, then select or unselect each resource. To activate all the resources, select All; you can also select None. By default, the value of the startResources property is None, so none of the predefined resources are activated.

❖ **Adding resources**

1. Expand the Resources folder.
2. Highlight the folder of the resource type you want to add, right-click, and select Add.
3. Complete the wizard to add the new resource, and click Finish.

Security tab

Process Identity

On UNIX platforms, EAServer allows you to configure an effective user name and group for the server to run as. This 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 to which the effective user belongs. 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:

- **UNIX User Name** The name of the effective user that runs the server process.

- **UNIX Group Name** A group to which the effective user belongs.

This feature is useful if you use listener ports less than 1024, such as 80 for HTTP and 443 for SSL. You cannot use port numbers less than 1024 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 properties are ignored on Windows platforms.

Secure Socket Options

The Java Secure Sockets Extension (JSSE) classes provide secure HTTP-tunnelled (HTTPS protocol) connections. JSSE provides an alternative to the built-in SSL implementations when secure connections are needed from an applet running in a Web browser. To use JSSE, configure these properties on the JSSE tab:

- **SSL Trust Store** The store that contains trusted certificates. The configuration property name is trustStore.
- **SSL Trust Store Password** The password to access the truststore. The configuration property name is trustStorePassword.
- **SSL Trust Store Type** The truststore type; for example, JKS or PKCS12. The configuration property name is trustStoreType.
- **SSL Key Store** The store that contains private keys and their associated public keys. The configuration property name is keyStore.
- **SSL Key Store Password** The password to access the keystore. The configuration property name is keyStorePassword.
- **SSL Key Store Type** The keystore type; for example, JKS or PKCS12. The configuration property name is keyStoreType.
- **FIPS Mode Enabled** Select to enable Federal Information Processing Standards (FIPS) mode. The configuration property name is fipsEnabled.

Additional configuration may be required—see “Configuring JSSE” in Chapter 10, “Security Configuration Tasks,” of the *EAServer Security Administration and Programming Guide*.

Services tab

On the Services tab, you can schedule service components to start automatically when the server starts.

Table 3-6 describes the installed service components, which you can schedule to start automatically when the server starts. To specify individual service components, choose Select, then select or unselect each service component. To schedule all the service components to start, select All; you can also select None. The configuration property serviceComponents defines the services to start, as a comma-separated list.

To add and configure individual service components, see "Adding and configuring service components" on page 39.

To create service components, see Chapter 4, "Creating Service Components," in the *EAServer Automated Configuration Guide*.

Table 3-6: Service components

Service component	Description
ConnectorWorkManager	Allows you to submit work to be performed synchronously or asynchronously. This component is used primarily by inbound connectors.
HelpServer	Implements help.
JaguarServer	Implements the CORBA container. If not started, the server runs in J2EE-only mode.
MBeanServer	Implements the JMX MBean server—see Chapter 14, "Systems Management."
SnmpService	Implements the JMX agent; depends on MBeanServer—see Chapter 14, "Systems Management."

Adding and configuring service components

❖ Adding service components

1. Highlight the Service Components folder, right-click, and select Add.
2. Complete the wizard to add the new service component, and click Finish.

❖ Configuring service components

1. Expand the Service Components folder, and select the service component to configure.
2. In the right pane, edit the service component properties defined in Table 3-7, then select Apply.

Table 3-7: Service component properties

Property	Description
Component	The name of the component that implements the service. For a user-defined service component, enter either: • The class name of a simple Java class (must be in the server class path), or • The fully-qualified name of an EJB local or remote interface; for example, ejb.components.mypackage.MyCompLocal or ejb.components.mypackage.MyCompRemote. The configuration property name is component.
Depends On	Select the service components on which this service component depends. The configuration property name is dependsOn.
Start Before Binding	Specify whether to start the server before name service bindings are set up. The configuration property name is startBeforeBinding.
Start Order	Specify the order in which to start the service component. The configuration property name is startOrder.
Run Thread Count	Specify the number of threads required to run the service component. The configuration property name is runThreadCount.
Stop Wait Time (ms)	Enter the number of milliseconds the server waits for the service to stop during server shutdown. If the service has not completely stopped when the time expires, the server shuts down anyway; the default is 1000 (1 second). The configuration property name is stopWaitTime.
Allow Start Failure	Select this option to continue if the service component fails to start. The configuration property name is allowStartFailure.

Tasks tab

The tasks tab allows you to schedule tasks to be performed automatically. For more information, see Chapter 3, “Using Scheduled Tasks,” in the *EAServer Automated Configuration Guide*.

Configuring listeners

A listener is an EAServer port that communicates with 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 and IIOP), no security profile is required. For information about security profiles, see "Configuring security profiles" in Chapter 10, "Security Configuration Tasks," in the *EAServer Security Administration and Programming Guide*.

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 (IIOP, and HTTP).

Preconfigured listeners

EAServer includes preconfigured listeners for the HTTP, IIOP, and RMI protocols. The default settings for the preconfigured listeners are described in Table 3-8. Only secure listeners use security profiles.

Table 3-8: Default listener settings

Listener name	Port	Security profile
http1	8000	
https1	8001	default
https2	8002	default_mutual
iiop1	2000	
iiops1	2001	default
iiops2	2002	default_mutual
rmi	1999	

The default host for these listeners is the name of the machine where EAServer is installed. The host is defined by the HOSTNAME environment variable. When you install EAServer, the installation process sets HOSTNAME to the name of your machine. To change the host for a listener, use the Management Console or jagtool. See "Configuring listener properties" on page 42, or the jagtool command set_props in Chapter 6, "Using jagtool and jagant," in the *EAServer Automated Configuration Guide*.

Listener failover

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 the Management Console to connect to the indicated IIOP address and reconfigure the server's listener properties.

Adding and configuring listeners

This section describes how to add, modify, and delete a listener.

❖ **Adding a new listener**

1. Highlight Listeners, right-click, and select Add.
2. Complete the wizard to create the new listener. Specify these values:
   - **Listener Name** A display name for the listener.
   - **Protocol** IIOP, IIOPS, HTTP, HTTPS, or RMI.
   - **Host Name** The name of the machine on which EAServer is running; the default is `${host.name}`.
   - **Port** The port number where the server listens for requests.
3. Select Finish.

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 "OS Configuration" on page 30.

❖ **Deleting a listener**

To disable a listener, see "Listeners tab" on page 30. To delete a listener configuration:

1. Expand the Listeners folder, then select the listener to delete.
2. Right-click, and select Delete.
3. In the wizard, verify that you want to delete the listener.

❖ **Configuring listener properties**

1. Expand the Listeners folder, then select the listener to modify.
2. The listener properties display in the right pane:

- General tab
- Performance tab

If you edit listener properties, select Apply to save your changes.

Note If you modify an existing listener, restart the server for your changes to take effect. If you change the server's host name or port number, enter the new URL in the Management Console and reconnect to the server.

General tab

On the listener's General tab, you can configure the following properties:

- **Protocol** IIOP, IIOPS, HTTP, HTTPS, or RMI. The protocol determines the type of clients that are supported. Protocols that end with "s" provide secure socket support.
  - HTTP and HTTPS – support browser and Web service clients.
  - IIOP and IIOPS – support EJB and CORBA clients.
  - RMI – supports JMX system management clients.

  The configuration property name is protocol.
- **Host Name** The name of the machine on which EAServer is running. The configuration property name is host.
- **Port** The port number where the server listens for requests. The configuration property name is port.
- **Listen Back Log** The maximum backlog for incoming requests. The configuration property name is listenBackLog.
- **Security Profile** If the protocol is HTTPS or IIOPS, enter the name of the security profile to use for this listener—see Chapter 10, "Security Configuration Tasks," in the *EAServer Security Administration and Programming Guide*. The configuration property name is securityProfile.
- **Include Server Info in Session ID** Include the server's IP address and port number in the HTTP session ID. This is required if you are using a Web redirector for load balancing. The configuration property name is includeServerInfoInSession.

- **Maximum Number of Threads** HTTP and HTTP listeners only. Specify the size of the thread pool that is used to run HTTP requests. This number can be less than the number of client connections. EAServer runs an HTTP request by pulling a thread from the pool to run the request. If the thread pool is too small, you may see `[SocketListener] LOW ON THREADS` errors in the log. This happens when too many clients try to send requests at the same time. The configuration property name is `maxThreads`. The default is -1, which indicates that EAServer uses the Jetty default value of 256.

Performance tab

On the Performance tab, you can configure performance at the listener level. See the property descriptions under "Monitors tab" on page 35.

Multiple listener hosts support

EAServer on UNIX platforms allows you to configure multiple host addresses for network listeners (URLs), which is useful when the machine is configured with multiple network interfaces. When specifying the URL, you can enter multiple URLs, separated by commas. For example:

```
http://host1:2000,http://host2:2000
```

You can also enter the 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 URL for each host on the specified port. If the server is in a cluster and you use 0.0.0.0, there is no localhost listener.

Monitoring threads

The primary purpose of thread monitors is to control resource use (for example, memory) by limiting the number of threads that can concurrently execute a particular task. Thread monitors also enable EAServer to detect whether a specific thread has been executing too long within a unit of work, which may indicate a thread that has stopped responding. A thread in this state can cause the system to become unresponsive. Thread monitors can check the managed threads in the system, such as Web container threads, but cannot monitor unmanaged threads, such as those created by applications. If EAServer detects a problem, an error is written to the server log file.

❖ **Adding a new thread monitor**

1. Expand Threads, select Thread Monitors, right-click, and select Add.
2. Complete the wizard to create the new thread monitor. Enter a name for the thread monitor.
3. Select Finish.

❖ **Monitoring active or waiting threads in the Management Console**

- Expand Threads, then select Active Threads or Waiting Threads.

❖ **Deleting a thread monitor**

1. Select the thread monitor, right-click, and select Delete.
2. In the wizard, confirm that you want to delete the thread monitor.

❖ **Configuring a thread monitor**

1. Expand the Threads folder, then expand the Thread Monitors folder, and select the thread monitor.

   The thread monitor properties display in the right pane on these tabs:

   - General tab
   - Advanced tab

   If you edit thread monitor properties, select Apply to save your changes.

General tab

On the General tab, configure these properties:

- **Allow Nesting** Allow thread monitors to be nested; by default, this is enabled. The configuration property name is allowNesting.
- **Bypass Domain** Select All, None, or choose Select, then specify which domains to bypass. Users in the specified domain can query the thread monitor for status, even if the monitor is at the thread limit. The configuration property name is bypassDomain.
- **Maximum Active Threads** The maximum number of active threads allowed; the default is 2147483647. The configuration property name is maximumActiveThreads.
- **Minimum Active Threads** The minimum number of active threads; the default is 0. The configuration property name is mimimumActiveThreads.

Advanced tab

On the Advanced tab, if you modify the value of Maximum Threads, select Apply, then select Synchronize to update the value of the maximumThreads property in the Ant configuration script that defines the thread monitor:

- **Maximum Threads** The maximum number of threads; the default is 0.

❖ Adding thread monitor properties

If you add configuration properties for a thread monitor, they are added to the configuration script. The next time you run the script, the properties are applied to the thread monitor.

1. Click Add Property.
2. Enter the property name and property value using this syntax:

   propertyName propertyValue.
3. Click Apply to save your changes, then click Synchronize to update the configuration file.

❖ Deleting thread monitor properties

1. Click Delete Property.
2. Click Apply, then click Synchronize.

Setting environment variables

In most cases, the EAServer installer sets the environment variables that are required to run the server. If your system requires configuration, use the methods described below.

❖ Configuring environment variables

1 Edit the *local-setenv.sh* (UNIX) or *local-setenv.bat* (Windows) file in the EAServer *bin* subdirectory.

  local-setenv.sh or *local-setenv.bat* is read when you start the server.

2 Set environment variables such as CLASSPATH, JAVA_HOME, and DJC_JVM_OPTIONS.

  By default, EAServer uses the JVM defined by the java command. To specify which JVM to use, set the DJC_JVM_OPTIONS environment variable to one of the following:

  - -server – HotSpot Server VM.
  - -client – HotSpot Client VM.

  You can also set DJC_JVM_OPTIONS using the Management Console—see Java Start-up Options in Table 3-1 on page 24.

3 To set Java logging options, see "Configuring system logging" on page 32.

4 Pass values directly to the Java VM by specifying arguments for DJC_USER_JVM_ARGS; for example, on UNIX:

```
DJC_USER_JVM_ARGS="-Dfoo=bar"
export DJC_USER_JVM_ARGS
```

  On Windows:

```
set DJC_USER_JVM_ARGS="-Dfoo=bar"
```

  You can also set these arguments on the command line.

Starting the server

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

Command line syntax

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:

1. By default, EAServer runs with JDK version 1.4. To run with JDK version 1.5, start the server using the -jdk15 option, or set the JAVA_HOME environment variable to the JDK 1.5 installation directory—see “Configuring environment variables” on page 47.

   Switching back to JDK 1.4 from JDK 1.5 Classes compiled with JDK 1.5 cannot be run in JDK 1.4 or earlier JDK versions. Since EAServer generates new classes for deployed applications, applications deployed with JDK 1.5 may not run if the server is restarted with JDK 1.4. To switch back to JDK 1.4 from JDK 1.5, undeploy and redeploy any applications that were deployed to the server running in JDK 1.5. Applications deployed to JDK 1.4 must also be compiled with JDK 1.4 (or with JDK 1.5 while specifying JDK 1.4 compatibility with the `-source 1.4` option to javac).

2. EAServer uses the JVM defined by the java command. To specify which JVM to use, set the DJC_JVM_OPTIONS environment variable to one of the following:

   - -server – HotSpot Server VM.
   - -client – HotSpot Client VM.

3. Change to the EAServer *bin* subdirectory.

4. Run either *start-server.sh* (UNIX) or *start-server.bat* (Windows) with the options described in Table 3-9.

Table 3-9: Server command line options

Option	Description
server	Specifies the name of the server; the default is the name of the machine on which EAServer is installed.
-bg	Windows only. Runs the server in the background. By default, the server runs in a command window.

Option	Description
-debug	Runs the server in debug mode. Running a server in debug mode allows you to remotely debug components from tools that support EAServer component debugging, such as JBuilder. **Note** The server cannot run in debug mode as a Windows service.
-jdk14	Starts the server with JDK 1.4.
-jdk15	Starts the server with JDK 1.5.
-jpda	Runs the server in Java Platform Debugger Architecture (JPDA) mode.
-jpdaSuspend	Used with -jpda, this option suspends the server at start-up time. This allows you to set breakpoints in code that executes at start-up, such as servlets that are configured to load on start-up. You can resume execution of the server with your Java debugger.The default port to use for JPDA debugging is 5005.
-ejbTrace	Traces EJB component invocations in the server log. To enable permanently (trace EJB components every time you run the server), set the ejbTrace server property—see "Log/Trace tab" on page 30.
-exRef	Logs exception cross-references, which is helpful for debugging.
-jmsTrace	Traces JMS operations in the server log. All public and protected JMS provider methods are traced. To enable permanently, set the jmsTrace server property—see "Log/Trace tab" on page 30.
-quiet	Runs the server without echoing messages.
-rmiTrace	Traces remote method invocations for IIOP and RMI-IIOP in the server log. To enable permanently, set the rmiTrace server property—see "Log/Trace tab" on page 30.
-rt14	Starts the server with Java runtime 1.4.
-rt15	Starts the server with Java runtime 1.5.
-sqlTrace	Traces JDBC driver activity in the server log, including JDBC prepared statement operations, parameter and result information for container-managed persistence operations (queries and updates), and transaction commit/rollback operations. To enable permanently, set the sqlTrace server property—see "Log/Trace tab" on page 30.

Option	Description
-txRef	Logs transaction cross-references, which is helpful for tuning server performance. For each query and transaction, “count” information is logged to a file in the *logs/transactions* directory. These logs allow you to determine what database interactions occur at runtime for EJBs using JDBC, entity beans using CMP, and MDBs. A new log file is created every five minutes.
-verbose	Enables verbose debugging.
-webTrace	Traces Web component invocations in the server log. To enable permanently, set the webTrace server property—see “Log/Trace tab” on page 30.

EAServer supports Internet Protocol Version 6 (IPV6). For more information, see “IPV6 support” on page 56.

Starting UNIX servers

Change to the EAServer *bin* directory, and run the *start-server.sh* script, using this syntax:

start-server.sh [*server*] [-jdk14 | -jdk15] [-rt14 | -rt15] [-debug] [-ejbTrace] [-exRef] [-jmsTrace] [-jpda] [-jpdaSuspend] [-quiet] [-rmiTrace] [-sqlTrace] [-txRef] [-verbose] [-webTrace]

See Table 3-9 on page 48 for a description of the server options.

Running UNIX servers in the background

To run a UNIX server in the background, change to the *$DJC_HOME/bin* directory, and run:

```
start-server.sh server &
```

where *server* is the server name as displayed in the Management Console.

To run the server in the background, in an xterm window, run:

```
start-server.sh server -xterm
```

Console output (stderr and stdout) is directed to the server log file.

Starting Windows servers

1. Change to the *%DJC_HOME%\bin* directory.
2. Run *start-server.bat* using this syntax:

   ```
   start-server [server] [-bg] [-jdk14 | -jdk15] [-rt14 | -rt15]
   [-debug] [-ejbTrace] [-exRef] [-jmsTrace] [-jpda] [-jpdaSuspend]
   [-quiet] [-rmiTrace] [-sqlTrace] [-txRef] [-verbose] [-webTrace]
   ```

   Table 3-9 on page 48 describes the options you can use to start a server.

Restarting servers

If you make changes to a server's configuration properties, restart the server for the changes to take effect.

1. Change to the server's *bin* directory.
2. To restart the local server, verify that the server's LocalRestart task is enabled (see "Tasks tab" on page 40), then run *restart-server.bat* (Windows) or *restart-server.sh* (UNIX):

   ```
   restart-server -local
   ```

   To restart another server, specify the name of the server:

   ```
   restart-server server
   ```

Stopping servers

You can shut down a server either using the Management Console (see "Shutting down a server" on page 6) or on the command line.

To stop a locally running server:

1. Change to the server's *bin* directory.
2. Verify that the server's LocalStop task is enabled (see "Tasks tab" on page 40), then run *stop-server.bat* (Windows) or *stop-server.sh* (UNIX):

```
stop-server -local
```

Note Locally running servers can also be stopped by killing the server process using the Task Manager (on Windows platforms) or the kill command (on UNIX and Linux platforms). Kill the server process only as a last resort; doing so may leave dependent services running such as the help server or database servers started from within EAServer. The server executable is java.exe (on Windows) or java (on UNIX and Linux).

To stop a server that is running on a remote machine:

1 Change to the server's *bin* directory.

2 Log in to the remote server using wlogin.

3 Run *stop-server.bat* (Windows) or *stop-server.sh* (UNIX), and specify the name of the server:

```
stop-server server
```

Running servers as Windows services

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 Configure any environment variables required for the server process. The DJC_HOME, PATH, CLASSPATH environment variables are saved in a configuration file when you install the server as a service.

2 Run the following command in the EAServer *bin* subdirectory, substituting the server name for *server* and the service name for *service*:

```
service [-server server] -install [-servicename service]
```

The environment variables and other service options are saved in *%DJC_HOME%\config\winservice_serviceName.ini*, where *serviceName* is the name of the service.

If you run `service -install` with no options, the local server is installed as a service called EAServer, and the name of the configuration file is *winservice_EAServer.ini*.

3 Configure the service to run using the account of the user who installed the service.

   On Windows 2000 and Windows XP:

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

   b Highlight the name of the service, right-click, and select Properties.

   c In the Properties dialog, select the Log On tab, then select Log On As This Account, and enter:

      - The account name of the user who installed the service. You can also click Browse, and select the account name.
      - The password for the user account.

   d Click Apply.

Note If you delete a server definition after installing the server as a Windows service, remove the server from the list of Windows services.

❖ Manually starting servers that are installed as Windows services

If a server is installed as a Windows service, the server starts 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. The name of the preconfigured server is the name of the machine on which EAServer is installed.

   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:

```
service -servicename serviceName -start
```

   where *serviceName* is the name of the server. To stop the server, run:

```
service -servicename serviceName -stop
```

Table 3-10 describes the options for managing Windows services.

Table 3-10: Windows service options

Option	Description
-config	Reconfigures an installed service. If you use the -config option to change system settings, such as CLASSPATH, the service settings that are stored in the Registry are reconfigured.
-debug	Allows you to test a service by running it on the command line. The service need not be installed. To simulate and test stopping, pausing, and continuing, start the service using the -debug option, then: • To toggle between pause and continue, use Ctrl+Break. • To stop the service, use Ctrl+C.
-install	Installs the server as a Windows service.
-remove	Removes the server from the list of Windows services.
-restart	Stops, then restarts the service.
-server *server*	The name of the server.
-servicename *service*	Specifies a service name. If not specified, the default is the server name.
-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.
-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 the Management Console.

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

- Run:

  ```
  service -servicename service -remove
  ```

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

Configuring the server license

Use the Management Console to configure the server license.

❖ **Setting server license properties**

1. Expand the License Configuration node, and select the server to configure.
2. On the General tab, set the following properties:
   - **License Product Edition** Select the product edition:
     - Advanced Edition
     - Developer Edition
     - Workgroup Edition
   - **License Log Level** Select the server logging level:
     - AUDIT – log auditing statements.
     - DEBUG – log debug statements.
     - ERROR – log all errors.
     - FATAL – log fatal errors only.
     - INFO – verbose logging.
     - TRACE – log tracing statements.
     - WARN – log warnings and errors.
     - NONE – no logging.
   - **License Type** Select the license type:
     - Application Deployment CPU License (AC)
     - Application Deployment Other License (AO)
     - Application Deployment Standby CPU License (BC)
     - Standby CPU License (SF)
     - Server License (SR)
     - Standalone Seat License (SS)
     - CPU License (CP)
     - Development and Testing License (DT)
     - Other License (OT)

     For more detailed information about the license types, see the *EAServer Installation Guide* for your platform.

- **Enable E-mail Properties** Enable sending error notifications using an e-mail application.
- **E-mail Properties** To send e-mail notifications when server errors occur, specify the e-mail properties:

E-mail property	Description
Severity	The minimum severity level that must exist before an e-mail notification is sent
Recipients	The e-mail addresses to which notifications are sent
Sender	The return e-mail address to which notification replies are sent
SMTP Host	The Simple Mail Transfer Protocol (SMTP) host to use for routing e-mail notifications
SMTP Port	The port number to use for contacting the SMTP host

3 To save your changes, select Apply. To restore the previous values, select Reset.

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 and higher. Windows 2000 does not support IPV6.

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/
```

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 data sources and J2EE connectors, and configuring Adaptive Server™ Enterprise user connections.

You can perform most configuration tasks using the Management Console.

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. 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 data source. The sample Java components connect using the Sybase jConnect for JDBC driver. View the predefined sample data sources in the Management Console for examples of configuring a data source 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 58.

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

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 JDBC, use the Management Console to define a JDBC data source that connects using the JDK 1.4 or later implementation of the jConnect driver. EAServer includes jConnect 6.05 runtime classes in *DJC_HOME/lib/ext/jconn3.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 Sybase Product Manuals Web site at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.jconnjdbc_6.05.prjdbc/html/prjdbc/title.htm&toc=/com.sybase.help.jconnjdbc_6.05/toc.xml.

jConnect metadata support required If you have transactional components that use jConnect data sources, 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 9 and 10. 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 data sources that use ODBC or JDBC as for any other ODBC or JDBC data source, as described in "Other databases" on page 59.

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

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 data source to use it.

To achieve JDBC connectivity, install the appropriate JDBC driver, then define an EAServer data source 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 configuration. For example, you may need to install stored procedures on each server to which you intend to connect. See your JDBC driver documentation for complete 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 data sources" on page 66 for information on defining data sources.

Configuring database types

Table 4-1 describes the predefined database types. Use the Management Console to modify the definitions of these database types and create other database types.

The predefined database types are configured in the *default-database-types.xml* Ant configuration file, located in the *config* subdirectory of your EAServer installation.

Table 4-1: Database types

Database type	Description
JCM_Odbc	An ASA database that uses the com.sybase.jaguar.jcm.odbc.OdbcDriver driver.
JCM_Odbc_Unicode	Adaptive Server Enterprise database that uses the com.sybase.jaguar.jcm.odbc.OdbcuDriver driver.
JCM_Oracle	Oracle database that uses the com.sybase.jaguar.jcm.oracle.OracleDriver driver. This database type corresponds to the EAServer 5.*x* OCI8, OCI9, and OCI10 native connection caches.

Database type	Description
JCM_Oracle_Unicode	Oracle database that uses the com.sybase.jaguar.jcm.oracle.OracleuDriver driver. This database type corresponds to the EAServer 5.*x* OCI8U, OCI9U, and OCI10U native connection caches.
JCM_Sybase	ASA database that uses the com.sybase.jaguar.jcm.sybase.SybaseDriver driver. This database type corresponds to the EAServer 5.*x* CTLIB_110 native connection cache.
Oracle	Oracle database that uses the oracle.jdbc.driver.OracleDriver driver.
Sybase_ASA	ASA database that uses the com.sybase.djc.sql.jit.SybaseDataSource3 driver EAServer uses an ASA database for Java Message Service storage, EJB container-managed persistence, EJB and Web session state storage, and the XA recovery log. To use another database type, reconfigure the default data source—see "Configuring data sources" on page 66.
Sybase_ASE	Adaptive Server Enterprise database that uses the com.sybase.djc.sql.jit.SybaseDataSource3 driver.

❖ Creating a new database type

1. In the Management Console, open the Resources folder.
2. Highlight Database Types, right-click, and select Add.
3. Enter the database name in the wizard.

   Database names must be one word, which can contain letters, numbers, and underscores. Names are case-sensitive. You cannot modify the name of an existing database.

Database entries appear in the right pane of the Management Console when you highlight the Database Types folder in the left pane.

❖ Deleting a database type

1. Expand the Database Types folder.
2. Highlight the database type to delete, right-click, and select Delete.
3. In the wizard, select Finish.

❖ **Configuring a database type**

1. Expand the Database Types folder.
2. Select the database type to configure.
3. In the right pane, enter the database type properties on these tabs:
   - General tab
   - Database Settings tab
   - SQL Settings tab

   To save your changes, select Apply. To restore the initial values, select Reset.

Note You must restart the server for changes to take effect.

General tab

Set these properties on the General tab:

- **JDBC Driver Class** The JDBC driver class name; for example, for a JCM_Sybase database type, enter:

  ```
  com.sybase.jaguar.jcm.sybase.SybaseDriver
  ```

  The configuration property name is driverClass.

- **JDBC Data Source Class** The class or library name used to support single-phase commit transactions. The configuration property name is dataSourceClass. For Sybase_ASA or Sybase_ASE database types, enter:

  ```
  com.sybase.jdbc3.jdbc.SybDataSource
  ```

  For an Oracle database type, enter:

  ```
  oracle.jdbc.pool.OracleDataSource
  ```

- **JDBC/XA Data Source Class** The class name or library name used to support two-phase commit transactions, and the name of the XA resource library. The configuration property name is xaDataSourceClass For Sybase_ASA or Sybase_ASE database types, enter:

  ```
  com.sybase.jdbc3.jdbc.SybXADataSource
  ```

  For an Oracle database type, enter:

  ```
  oracle.jdbc.xa.client.OracleXADataSource
  ```

- **Ping Connection SQL Statement** SQL statement to use when testing the database connection with ping; the default is “select 1”. The configuration property name is pingSQL.
- **Driver Debugging On/Off** Select to enable debugging for the driver; unselect to disable. The configuration property name is driverDebug.
- **Driver Debugging Settings** Debug settings for the driver debugger. The configuration property name is driverDebugSettings, and the default is “STATIC:ALL.”
- **Truncate Nanoseconds** A divisor/multiplier that is used to round the nanoseconds value in a java.sql.Timestamp to a granularity that the DBMS supports. The configuration property name is truncateNanos, and the default is 10000000.
- **Maximum Length for BLOB Update** The maximum number of bytes allowed when updating a BLOB datatype using PreparedStatement.setBytes. The configuration property name is psMaximumBlobLength, and the default is 16384.
- **Maximum Length for CLOB Update** The maximum number of characters allowed when updating a CLOB datatype using PreparedStatement.setString. The configuration property name is psMaximumClobLength, and the default is 16384.

Database Settings tab

Configure these properties on the Database Settings tab:

- **Default Database URL** The JDBC URL for connecting to the database. The configuration property name is databaseURL. For a Sybase_ASA database, enter:

  ```
  jdbc:sybase:Tds:${serverName}:${portNumber}
  ```

  For other database types, see the value of databaseURL in *config/default-database-types.xml* in your EAServer installation.
- **Default Database Name** The default database name for this database type. The configuration property name is databaseName. For a Sybase_ASA database, the default database is *default.db*.
- **Default Network Protocol** The network protocol that the database server uses for network communication. The configuration property name is networkProtocol.

- **Default Server Name** The name of the host where the database server is running. The configuration property name is serverName.
- **Default Port Number** The server port number where the database server listens for connection requests; typically, a TCP/IP port number 1 – 65535. The configuration property name is portNumber.
- **Default User** The user name for connecting to the database. The configuration property name is user. The default user name for a Sybase_ASA database is "dba."
- **Default Password** The password for connecting to the database. The configuration property name is password. The default password for the dba user in a Sybase_ASA database is "sql."
- **Default Role Name** The default role name for data sources that access the database type.The configuration property name is roleName.
- **Default Service Name** The default service name for data sources that use this database type. The configuration property name is serviceName.

SQL Settings tab

Configure these properties on the SQL Settings tab:

- **Use Quoted Identifiers** If you select this option, SQL identifiers are quoted. The configuration property name is useQuotedIdentifiers.
- **Disable Prefetch** Prefetch optimizes container-managed persistence by batching queries from a parent to its children (for example, customer to orders), to reduce the calls from the application server to the database. The configuration property name is disablePrefetch.
- **Batch Delimiter** A delimiter; for example, a semicolon, which can be used to separate multiple SQL statements within a statement batch. The configuration property name is batchDelimiter.
- **BLOB Updater** The name of a class that can be used for updating database BLOB (long binary) objects when the BLOB size is greater than psMaximumBlobLength. The class must implement the com.sybase.djc.sql.BlobUpdater interface. The configuration property name is blobUpdater. A sample implementation for Oracle is documented in the *html/en/com.sybase.djc.sql.DatabaseType.html* file, in your EAServer installation.

- **CLOB Updater** The name of a class that can be used for updating database CLOB (long string) objects when the CLOB size is greater than psMaximumClobLength. The class must implement the com.sybase.djc.sql.ClobUpdater interface. The configuration property name is clobUpdater. For an Oracle sample implementation, see *html/en/com.sybase.djc.sql.DatabaseType.html*, in your EAServer installation.
- **After Insert SQL** If the database requires "insert into" for database inserts, instead of the abbreviated form "insert," enter `into`. The configuration property name is afterInsert.
- **Check Update SQL** Not currently used.
- **Check Delete SQL** Not currently used.
- **Compact Column Alias Prefix** An expression that uses the nested variables "${index}" and "${column}" for shortening column names in result sets. This can reduce the data transmitted between the database server and the application server. For example:

  ```
   _${index}=${column}
  ${column} AS _${index}
  ```

  The configuration property name is compactColumnAlias.
- **Select With Shared Lock (SQL Template)** A template SQL statement for selecting rows and acquiring a shared lock. The configuration property name is selectWithSharedLock. If your database server does not support shared locks, specify a template for acquiring exclusive locks. For example, the following is defined for the Sybase_ASA database type:

```
${selectList}${intoClause}${fromClause} holdlock${whereClause}
```

- **Select With Update Lock (SQL Template)** A template SQL statement for selecting rows and acquiring an exclusive lock. The configuration property name is selectWithUpdateLock. If your database server does not support exclusive locks, specify a template for acquiring shared locks. For example, the following is defined for the Sybase_ASA database type:

```
update ${mainTable} set ${touchColumn} =
1 -${touchColumn}${fromClause}${whereClause};;
${selectList}${intoClause}${fromClause}${whereClause}
```

- **Serializable Select (SQL Template)** A template SQL statement for selecting rows and acquiring a lock that ensures strict serializability, in terms of equivalence with serial schedules. For example, for a Sybase database:

```
${selectList}${intoClause}${fromClause} holdlock${whereClause}
```

. The configuration property name is serializableSelect.

Note Some databases do not provide the syntax to acquire this type of lock (for example, Oracle). Therefore, it is impossible to support the serializable persistent object isolation level for all database types.

Advanced tab

The Advanced tab lists all the properties and values that are defined in the database type's Ant configuration script. If you change any values on this tab, click Apply to save your changes, then click Synchronize to update the values in the Ant configuration script.

Configuring data sources

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

If you define two data sources that use identical values for server, user, and password, and your application requests a connection using only these values, EAServer returns the first match that is found.

Table 4-2 describes the predefined data sources. Use the Management Console to modify the definitions of these data sources and create new data sources.

Table 4-2: Predefined data sources

Data source name	**Description**
default	The default data source, which is configured to connect to a Sybase_ASA database type.
cluster.db	Initially, the same configuration as the default data source; you may want to customize to use with a server cluster.
JavaCache	An alias for default.
message.db	Initially, the same configuration as the default data source; you may want to customize to use for the message service.
session.db	Initially, the same configuration as the default data source; you may want to customize to use for a session.
tx_manager	Initially, the same configuration as the default data source; you may want to customize for the transaction manager.

❖ Creating a new data source

1. In the Management Console, open the Resources folder.
2. Highlight Data Sources, right-click and select Add.
3. Enter the data source name in the wizard.

   Data source names are limited to one word, which can contain letters, numbers, and underscores. Names are case-sensitive.

Data source entries appear on the right side of the window in the Management Console when you highlight the Data Sources folder on the left side of the window.

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

❖ Deleting a data source

1. Expand the Data Sources folder.
2. Highlight the data source to delete, right-click, and select Delete.
3. In the wizard, select Finish.

❖ Configuring a data source

You cannot modify the name of an existing data source.

1. Expand the Data Sources folder.

2. Select the data source to configure.
3. In the right pane, enter the data source properties on these configuration tabs:
   - General tab
   - Connection Pool tab
   - Database Settings tab
   - Data Source Driver tab
   - Advanced tab

   To save your changes, select Apply; to restore the initial values, select Reset.

General tab

The properties that you configure on the General tab are:

- **Alias For** To set up this data source to be an alias for another data source, enter the name of the other data source. When a data source name is specified, all other properties for this data source are ignored. Initially, each of the predefined data sources—cluster.db, JavaCache, message.db, session.db, and tx_manager—is configured as an alias for the default data source. The configuration property name is aliasFor.
- **Proxy For** This property is not used by EAServer.
- **Database Type** The database type. Required for data sources that use the EAServer automatic persistence or stateful failover features, such as EJB CMP entity beans. This property defines database-specific information that is required by the storage component, for example, the commands to verify that a table exists and to create new tables. The configuration property name is databaseType. The predefined database types are listed in Table 4-1 on page 60.
- **Database URL** The JDBC URL for connecting to the database. The configuration property name is databaseURL. You can reference other data source properties in the URL; for example:

  ```
  jdbc:sybase:Tds:${serverName}:${portNumber}
  ```

  If you specify “[default]” as the URL, the value of the database property Default Database URL is used.

- **Database Name** The name of the database for this data source. The configuration property name is databaseName. Once defined, you can refer to this as "${databaseName}" in other property definitions for this data source. If you specify "[default]" as the database name, the value of the database property Default Database Name is used.
- **Network Protocol** The network protocol that the database server uses for network communication. The configuration property name is networkProtocol. If the value is "[default]," the Default Network Protocol database property value is used.
- **Server Name** The name of the host where the database server is running; typically, a TCP/IP host name. The configuration property name is serverName. If you enter "[default]" as the server name, the value of the Default Server Name database property is used.
- **Port Number** The server port number where the database server listens for connection requests; typically, a TCP/IP port number 1 – 65535. The configuration property name is portNumber. If the value is "[default]," the Default Port Number database property value is used.
- **User** The name used (along with a password) to connect to the database identified by the server entry. The configuration property name is user.
- **Password** The password used to connect to the database identified by the server entry. The Management Console stores passwords in encrypted form. If you define the data source password using an XML configuration script, specify "*" to avoid having plain-text passwords stored in the data source properties file. The configuration property name is password. If the value is "[default]," the Default Password database property value is used.

  The Management Console does not display passwords for existing data sources. To change a password, enter the new password and click Apply.
- **Service Name** The service name for the data source. The configuration property is serviceName.
- **Locale/Codeset** The value of CS_SYB_CHARSET for this data source. This is currently applied only to Open Client connections with a JCM_Sybase database type. If the value is "[server]," the value of the current application server's defaultCodeSet property is used. The configuration property name is codeSet.
- **Locale/Language** The value of CS_SYB_LANG for this data source. This is currently applied only to Open Client connections with a JCM_Sybase database type. The configuration property name is language.

- **High Availability** Select to initialize CTLIB connections with the CS_HAFAILOVER attribute. The configuration property name is highAvailability.
- **Role Name** The database role that the user must have to log in to the database. The configuration property name is roleName. If the value is "[default]," the Default Role Name database property value is used—see "Database Settings tab" on page 63.
- **Owner Prefix** The owner prefix for stored procedures and table names in this data source. A prefix is used by the EJB persistence manager and JIT driver wrappers to qualify database identifiers for stored procedures and tables. The configuration property name is ownerPrefix.
- **Set Session Authorization** To establish an effective database identity that matches the current application server user, select to use an ANSI SQL set session authorization command at the start of each database transaction. The configuration property name is setSessionAuth.
- **Set Session Authorization System ID** If Set Session Authorization is selected, specify the database identity to use when the application server accesses the database from a transaction that runs with "system" identity. The configuration property name is setSessionAuthSystemID.
- **Commit Protocol** Specify how connections for this data source are handled at commit time. The configuration property name is commitProtocol:
  - Optimistic – enables connections to be committed without regard for other connections enlisted in the transaction, assuming the transaction is not marked for rollback and can successfully commit on all resources. This is the default.
  - XA_2PC – XA two-phase commit.

Note If you are using two-phase commit, the recovery log is stored in the tx_manager data source, and its commit protocol must be optimistic. If tx_manager is aliased to another data source, the commit protocol for that data source must be optimistic. A last-resource optimization is used to ensure full conformance with the XA specification. The commit protocol for all other data sources should be XA_2PC. Alternately, a transaction that accesses multiple data sources whose commit protocols are optimistic is permitted, but atomicity is not guaranteed.

- **Ping Pooled Connections** Select to ping connections before attempting to reuse them from the connection pool. Selecting this option somewhat degrades performance. If transaction retry is enabled for your EJB transactions, dead database connections are discarded automatically (usually after a rollback), and you can improve performance by unselecting this property. The configuration property name is pingConnections.
- **Disable Prefetch** Prefetch optimizes container-managed persistence by batching queries from a parent to its children, to reduce the calls from the application server to the database. The configuration property is disablePrefetch.
- **Disable Triggers** Select to deactivate database triggers, on a per-connection basis, when the application server accesses the database. If selected, the database must support both the set triggers on and set triggers off commands. For Adaptive Server Enterprise, the data source user must have the replication_role database role.

  Disabling triggers is useful when using container-managed persistence with version columns for optimistic concurrency control. Database triggers permit external updates to version columns, by non-application-server clients. Disabling triggers prevents concurrent updates to version columns, which could lead to optimistic concurrency failures, and require transaction retries. This is particularly relevant if using an isolation level of ReadCommittedWithCache, RepeatableReadWithCache, or SerializableWithCache—see ejb.isolationLevel in “Commonly configured properties” in Chapter 2, “Deploying and Configuring EJB Components,” in the *EAServer Enterprise JavaBeans User’s Guide*.

Connection Pool tab

Configure these properties on the Connection Pool tab:

- **Initial Pool Size** The initial number of connections in the pool. EAServer preallocates and opens the specified number of connections at start-up time. The default value of initialPoolSize is zero.
- **Minimum Pool Size** The minimum number of connections in the pool. When open connections are idle, the pool is pruned to this size. The configuration property name is minPoolSize, and the default value is zero.

  If no minimum size is specified, connections are opened on an as-needed basis to fill the pool up to the maximum size.

- **Maximum Pool Size** The maximum number of connections allocated to the pool for this data source. If the maximum is exceeded, and cannot be resolved by waiting; for example, if deadlock occurs, you may see "ResourceMonitorTimeoutException" in your log file, and some transactions may fail. A value of zero sets no limit to the connection pool size. The configuration property name is maxPoolSize.
- **Maximum Idle Time** Specifies the number of seconds an idle connection remains in the pool before it is dropped. The default is 60 seconds. If the value is zero, idle connections remain in the pool until the server shuts down. The configuration property name is maxIdleTime.
- **Maximum Wait Time** The maximum number of seconds to wait for a connection before the request is cancelled. The configuration property name is maxWaitTime., and the default value is 60.
- **Maximum Cached Statements** The maximum number of JDBC prepared statements that can be cached for each connection by the JDBC driver. The value of this property is JDBC-driver specific. The configuration property name is maxStatements; the default is zero.

Database Settings tab

Configure these properties on the Database Settings tab:

- **Database File** The name of the database file for this data source. The configuration property name is databaseFile. You can reference this value as "${databaseFile}" in other data source property definitions. For a Sybase_ASA database, enter:

  ```
  ${djc.home}/data/default.db
  ```
- **Database Create Command** The operating system command used to create the database for this data source; for example:

  ```
  ${djc.home}/bin/asa-init${.bat} -q ${databaseFile}
  ```

  If this command is defined and the file referenced by `${databaseFile}` does not exist, the command is run to create the database when an application component attempts to obtain the first connection from the connection pool for this data source. The configuration property name is databaseCreateCommand.
- **Database Start Command** The operating system command used to start the database for this data source; for example:

```
${djc.home}/bin/asa-start${.bat} -x tcpip(localonly=yes) -n
```

```
${dataSource} ${databaseFile}
```

If this command is defined and the database is not running, the command is run to start the database when the data source is activated. The configuration property name is databaseStartCommand.

- **Database Stop Command** The operating system command used to stop the database for this data source. For a Sybase_ASA database, where the user name and password are the defaults (dba and sql), enter:

```
${djc.home}/bin/asa-stop${.bat} -y -c "uid=dba;pwd=sql"
```

If this property (databaseStopCommand) is defined and the database is running, this command executes when EAServer shuts down.

- **Database Command Echo** Select to echo the database commands databaseCreateCommand, databaseStartCommand, and databaseStopCommand in both the server console and the server log when the commands are run. The name for this configuration property is databaseCommandEcho.
- **Start Wait Time** If a database start command is specified, the number of seconds to wait for the start command to run before reporting a connection problem. If the start command completes successfully within this time period, no exceptions are reported in the server log. The configuration property name is startWait. The default is 60 seconds.

❖ **Deleting a data source**

1. Expand the Data Sources folder.
2. Highlight the data source to delete, right-click, and select Delete.
3. In the wizard, select Finish.

Data Source Driver tab

Configure these properties on the Data Source Driver tab:

- **JDBC Driver Class** The name of the class that implements the JDBC driver. The configuration property name is driverClass.
- **JDBC Data Source Class** The name of the class that implements the JDBC data source. The configuration property name is dataSourceClass.
- **JDBC/XA Data Source Class** The name of the class that implements the JDBC data source for XA (two-phase commit) connections. The configuration property name is xaDataSourceClass.

Advanced tab

On the Advanced tab, you can add property definitions, and synchronize changes with the Ant configuration script.

❖ Adding configuration properties

1. Click Add Property.
2. Enter the property definition as *propertyName* and *propertyValue*.
3. Click Apply to save your changes.
4. Click Synchronize to update the Ant configuration script.

Refreshing and testing data sources

If you modify data source properties, refresh the data source before you test it.

❖ Refreshing a data source

1. Expand the Data Sources folder.
2. Highlight the data source, right-click, and select Refresh.

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

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

❖ Testing a data source

1. Expand the Data Sources folder.
2. Highlight the data source, right-click, and select Ping.

   A message box notifies you if ping is successful.

Configuring connectors

A connector is a specialized connection factory that provides connections for EJBs, Java servlets, JSPs, and CORBA-Java components. Connectors are also called resource adapters.

EAServer 6.0 supports the J2EE Connection Architecture (JCA) 1.5 specification. JCA 1.5 connectors include support for message listeners, which can listen for any message type.

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 141. To undeploy a connector, see undeploy on page 197.

❖ **Configuring a connector**

- Expand the Connector Modules folder, then highlight the connector you want to configure. The connector properties display on:
  - General tab
  - ra.xml tab
  - Configuration tab
  - Advanced tab

  On each tab, click Apply to save your updates.

General tab

On the General tab, enter:

- **Resource Adapter Class** The fully qualified name of the class that implements the connector; for example, easqa.test.jca15.connector.deployment.ResourceAdapterImpl. The configuration property name is resourceAdapterClass.
- **Transaction Support** Select the transaction type that this connector supports—see "Transaction modes" on page 75. The configuration property name is transactionSupport.
- **Authentication Mechanisms** Enter the names of the supported authentication mechanisms, in a comma-separated list. The configuration property name is authenticationMechanisms.
- **Security Permissions** Enter the security permissions that are required to access the connector, in a comma-separated list. The configuration property name is securityPermissions.
- **Reauthentication Support** Select to enable support for reauthenticating existing managed connection instances. The configuration property name is reauthenticationSupport
- **Administered Objects** Enter the names of administered objects used by this connector, as a comma-separated list. The configuration property name is adminObjects.

ra.xml tab

The ra.xml tab displays the deployment descriptor for the connector (resource adapter) module. If you update the file, select Apply to save your changes.

Configuration tab

The Configuration tab displays the Ant configuration script that defines this connector.

❖ **Changing connector properties**

To change connector properties, edit the user configuration file, rather than the Ant configuration script.

1 If you see a User Configuration tab, select it; otherwise:

   a In the left pane, highlight the connector, right-click, and select Create User Configuration.

b. Highlight the connector again, right-click, and select Refresh Node. A user configuration script is created and displays in a new User Configuration tab. Select the tab.

2. Add properties and values you want to change. You can copy property definitions from the Ant configuration script, then edit the values. If you reconfigure or recompile the connector, the property values in the user configuration script override the property values in the Ant script.

   For information about creating and configuring Ant scripts and user configuration scripts, see Chapter 2, "Ant-Based Configuration," in the *EAServer Automated Configuration Guide*.

3. Recompile the connector. In the left pane, highlight the connector, right-click, and select Run Ant Recompile.

4. Refresh the server.

Advanced tab

On the Advanced tab, you can edit the following property values. If you make changes, select Synchronize to update the corresponding Ant configuration script:

- **config-prop:RAName** The name of the resource adapter.
- **djcClassLoader** The name of the custom class loader, which reloads classes when you refresh the connector. If not specified, classes are loaded by the Java system class loader, which means you must restart the server to use new versions of the classes.

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

❖ **Refreshing a connector**

1. Expand the Resource Adapters folder.
2. Select the connector you want to refresh, right-click, and select Refresh.

❖ **Refreshing connector views**

1. To refresh the tree view for all connectors, select the Resource Adapters folder.

   To refresh the tree view for a single connector, expand the Resource Adapters folder, and select the connector.

2 Right-click, and select Refresh Node.

❖ **Synchronizing a connector**

- You can synchronize a connector within a cluster of servers by using the Management Console to create an export configuration—see Chapter 7, “Exporting Server Modules.”

Authentication mechanisms

To configure the authentication method for a connector:

1 Expand these folders: Connector Modules | *Connector* | Authentication Mechanisms, where *Connector* is the name of the connector.

2 Select the authentication mechanism, and configure the properties that display on:

   - General tab
   - Advanced tab

General tab

On the General tab for an authentication mechanism, enter:

- **Mechanism Type** The authentication type; for example, `BasicPassword`. The configuration property name is mechanismType.
- **Credential Interface** The fully-qualified name of the interface class that represents the credential; for example, `javax.resource.spi.security.PasswordCredential`. The configuration property name is credentialInterface.

Advanced tab

On the Advanced tab for an authentication mechanism, you can edit the following property values. If you change either value, select Apply, then select Synchronize to update the corresponding Ant configuration script:

- **djcClassLoader** The name of the custom class loader, which reloads classes when you refresh the connector. If not specified, classes are loaded by the Java system class loader, which means you must restart the server to use new versions of the classes.

- **resourceAdapter** The name of the connector to which this authentication mechanism applies.

Administered objects

To configure a connector's administered objects:

1. Expand these folders: Connector Modules | *Connector* | Administered Objects, where *Connector* is the name of the connector.
2. Select the administered object, and configure the properties that display on:
   - General tab
   - Advanced tab

General tab

On the General tab, you can configure the following administered object properties:

- **Resource Adapter** Select the name of the resource adapter that administers this object. The configuration property name is resourceAdapter.
- **Administered Object Interface** The fully-qualified name of the class that implements the object interface; for example, `javax.jms.Queue`. The configuration property name is administeredObjectInterface.
- **Administered Object Class** The fully-qualified name of the class that implements the object; for example, `com.wombat.connector.jms.QueueImpl`. The configuration property name is administeredObjectClass.

Advanced tab

On the Advanced tab, you can specify the class loader. If you make changes, click Apply, then Synchronize to update the configuration script:

- **djcClassLoader** The name of the custom class loader, which reloads classes when you refresh the connector. If not specified, classes are loaded by the Java system class loader, which means you must restart the server to use new versions of the classes.

Connection factories

For each connector, one connection factory is created automatically, and the connection factory type is defined. You can add and configure connection factories, and configure connection factory types.

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();
```

❖ **Adding a connection factory**

To add connection factories for a connector:

1 Expand the Connector Modules folder.

2 Select the connector to which you want to add a connection factory, right-click, and select Add.

3 Complete the wizard to add the connection factory.

❖ **Configuring connection factory properties**

1 Expand these folders: Connector Modules | *Connector* | Connection Factories, where *Connector* is the name of the connector.

2 Select the connection factory, and configure the properties that display on:

- General tab
- Advanced tab

General tab

On the General tab, select:

- **Connection Factory Type** The name of the connection factory type. The configuration property name is connectionFactoryType.

Advanced tab

On the Advanced tab, you can specify the class loader. If you change the value, select Apply, then select Synchronize to update the Ant configuration script that defines the connection factory:

- **djcClassLoader** The name of the custom class loader, which reloads classes when you refresh the connector. If not specified, classes are loaded by the Java system class loader, which means you must restart the server to use new versions of the classes.

Connection factory types

When you deploy a connector, its connection factory types are defined. You can update the type definitions.

❖ **Configuring connection factory types**

1. Expand these folders: Connector Modules | *Connector* | Connection Factory Types, where *Connector* is the name of the connector.
2. Select the connection factory type, and configure the properties that display on:
    - General tab
    - Advanced tab

General tab

On the General tab, you can configure these connection factory type properties:

- **Resource Adapter** Select the connector from the list of those defined. The configuration property name is resourceAdapter.
- **Show Full Definition** Select to display all the configuration properties for the connection factory type.
- **Managed Connection Factory Class** The fully-qualified name of the class that implements the managed connection factory type. The configuration property name is managedConnectionFactoryTypeClass.
- **Connection Factory Interface** The name of the class that implements the connection factory interface. The configuration property name is connectionFactoryInterface.

- **Connection Factory Class** The fully-qualified name of the class that implements the connection factory. The configuration property name is connectionFactoryClass.
- **Connection Interface** The fully-qualified name of the class that implements the connection interface. The configuration property name is connectionInterface.
- **Connection Class** The fully-qualified name of the class that implements connection management. The configuration property name is connectionClass.

Advanced tab

On the Advanced tab, you can configure the following connection factory type properties. If you make changes, click Apply, then Synchronize to update the Ant configuration script:

- **connectionClass** The fully-qualified name of the class that implements connection management.
- **djcClassLoader** The name of the custom class loader, which reloads classes when you refresh the connection factory type. If not specified, classes are loaded by the Java system class loader, which means you must restart the server to use new versions of the classes.
- **managedConnectionFactoryClass** The fully-qualified name of the class that implements the managed connection factory.

JCA 1.5 message listeners

JCA 1.5 connectors can include **inbound resource adapters**, which are also called message listeners. You can receive messages through a connector by mapping its inbound resource adapter to a message-driven bean (MDB).

The Java Message Service (JMS) defines MDBs that listen for JMS messages. EJB 2.1 adds support for generalized MDBs, which can listen on a JCA 1.5 inbound resource adapter for any message type. Inbound resource adapters also enable MDBs to listen for messages from third-party JMS providers. See "Message-driven beans" in Chapter 2, "Setting up the Message Service," in the *EAServer Java Message Service User's Guide*.

Each JCA message listener is defined by its message listener type. You can configure a message listener type, then assign it to a message listener.

❖ **Configuring JCA 1.5 message listener types**

1. Expand these folders: Connector Modules | *Connector* | Message Listener Types, where *Connector* is the name of the connector.
2. Select the message listener type, and configure the properties that display on:
    - General tab
    - Advanced tab

General tab

On the General tab for the message listener type, enter:

- **Resource Adapter** Select the connector (resource adapter) from the list. The configuration property name is resourceAdapter.
- **Show Full Definition** Select to display all the configuration properties for the message listener type.
- **Message Listener Interface** The fully-qualified name of the class that implements the message listener interface. The configuration property name is messageListenerInterface.
- **Activation Specification Class** The fully-qualified name of the class that implements the activation specification. The configuration property name is activationSpecificationClass.

Advanced tab

On the Advanced tab for the message listener type, you can edit the following property values. If you make changes, click Apply, then Synchronize to update the corresponding Ant configuration script:

- **djcClassLoader** The name of the custom class loader, which reloads classes when you refresh the message listener type. If not specified, classes are loaded by the Java system class loader, which means you must restart the server to use new versions of the classes.
- **requiredConfigProperties** Any property with the name config-prop:*XXX* that is used to initialize the message listener's JavaBean-style property named *XXX*.

❖ **Configuring JCA 1.5 message listeners**

1. Expand these folders: Connector Modules | *Connector* | Message Listener Types, where *Connector* is the name of the connector.
2. Select the message listener, and configure the properties that display on:
   - General tab
   - Advanced tab

General tab

On the message listener General tab, select:

- **Message Listener Type** Select the message listener type that defines this message listener. The configuration property name is messageListenerType.

Advanced tab

On the message listener Advanced tab, you can edit the following property value. If you make changes, click Apply, then Synchronize to update the corresponding Ant configuration script:

- **djcClassLoader** The name of the custom class loader, which reloads classes when you refresh the message listener. If not specified, classes are loaded by the Java system class loader, which means you must restart the server to use new versions of the classes.

Receiving messages from a JCA 1.5 connector

To receive messages from a JCA 1.5 connector, map the inbound resource adapter to an MDB. In the Ant configuration file that defines the MDB, set the value of messageListener name to the name of the com.sybase.djc.connector.MessageListener component that implements the inbound resource adapter. In the following example, “myMDB” is the name of the MDB component, and “connectorListener” is the name of the inbound resource adapter:

```
<setProperties component="ejb.components.abctest.myMDB">
   <messageListener name="connectorListener" />
  </setProperties>
```

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 that number 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 information about configuring user connections, see these Adaptive Server Enterprise books on the Sybase Product Manuals Web site:

- Configuration Guide for UNIX at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_12.5.1.uconfig/html/uconfig/title.htm
- Configuration Guide for Windows at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_12.5.1.ntconfig/html/ntconfig/title.htm

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 operates its own independent name service. In an EAServer cluster, each cluster member knows the connection information of all the other members. Therefore, an EAServer client can choose any cluster members as name servers.

Using the Management Console, you can set naming service options for EJB and JMS providers. If an EJB or JMS provider is defined in the repository of a client’s EAServer installation, the client can use the URLs ejb-provider:*name* and jms-provider:*name*, where *name* is the name of the provider. You can also use named JMS providers for JMS “store and forward” (pull and push).

EAServer initial context

When you deploy an EJB module, you can set the ejb.remoteNamePrefix and ejb.remoteNameSuffix properties in the module's Ant configuration script to configure name prefix and suffix patterns. Deployment also creates default bindings of remote and home interfaces using the *PackageName/CompName* pattern. You can use package-level bind calls to bind arbitrary names to home interfaces.

Name bindings

Name bindings are initially defined in Ant configuration files. When the Ant files are run, the bindings are copied into component and package property files, which are distributed during cluster synchronization. EAServer loads name bindings into memory when a deployed module is started or refreshed.

Initially, the Ant configuration files contain settings read from the EJB-JAR deployment descriptor. You can customize properties, such as bindings, in a user-configuration file for an EJB module or Web application. For example, you can bind a name to a data source, and you can bind a data source to all the resource references in a package. See "Commonly configured properties" in Chapter 2, "Deploying and Configuring EJB Components," in the *EAServer Enterprise JavaBeans User's Guide*.

For details on working with Ant configuration files, see `Chapter 2, "Ant-Based Configuration,"` in the *EAServer Automated Configuration Guide*.

❖ **Viewing a server's name bindings**

1. Expand the Servers folder, and select the server.
2. Highlight the Name Bindings icon. The window in the right pane displays:
   - Bind Name – the name you can use to access an object.
   - Bind Object – the object to which the bind name refers.
   - Scope – the scope (client, cluster, server, package, or component) in which the name binding is valid.
   - Status – the status of the name binding; active or inactive.

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.

The EAServer JNDI implementation includes the JNDI service provider interface (SPI), which enables you to use a variety of custom directory and naming services.

Beginning in version 6.0, EAServer supports JNDI for EJB and JMS clients, and when required, for J2EE compliance. JNDI is no longer supported for CORBA clients, which should use the SessionManager CORBA interfaces. See the *EAServer CORBA Components 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, you can 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();
```

For information about using the Management Console to add and configure environment properties for a Web application, application client, or EJB component, see "Environment properties" in Chapter 1, "Defining Web Applications," in the *EAServer Web Application 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 the Management Console to add and configure EJB references in Web applications, EJB components, and application clients, see "EJB references" in Chapter 1, "Defining Web Applications," in the *EAServer Web Application 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 the Management Console to add and configure resource references in Web applications, EJB components, or application clients, see "Resource references" in Chapter 1, "Defining Web Applications," in the *EAServer Web Application 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();
```

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 8, "Load Balancing, Failover, and Component Availability." A *cluster partition* is a subset of a cluster, which identifies preferred and alternate servers, and can be assigned to service specific components' requests. *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 way to distribute information between servers.

Cluster overview

Each cluster includes a primary server, and a group of participating 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*, or nonprimary servers, must have unique names.
- A cluster consists of at least two servers. Each instance of EAServer operates its own independent name service, and each cluster member knows the connection information for all the other members. See "How does the EAServer naming service work?" on page 87.

- When you configure a cluster, you can define cluster partitions, which determine where remote clients using lookup can find components. To ensure high availability, a cluster partition must include at least two physical servers. For load balancing, a cluster partition may contain only one server.

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.

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.

Typically, each server in a cluster runs on a different host, so each server has its own copy of the entire repository and all the 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 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 a cluster uses PowerBuilder components, all the servers must use the same version of the PBVM.

Cluster servers

Each server can be a member of only one cluster. To provide high availability, there must be at least two servers defined for a cluster.

If you are not using partitions, add a new machine to a cluster by changing the host in each listener to the Internet host name or IP address. Then, connect the Management Console to the cluster’s primary server and synchronize the cluster—see Chapter 7, “Exporting Server Modules.”

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 either “localhost” or the machine name.

Creating and configuring clusters

This section describes how to use the Management Console to create, configure, and manage a cluster.

❖ **Creating a cluster**

1. Highlight the Clusters folder, right-click, and select Add.
2. In the New Cluster wizard, enter a name for the cluster.
3. Select the primary server. The default selection is the server to which the Management Console is connected.

   When you synchronize the cluster, the primary server's properties are distributed to the participating servers.
4. Select the participating servers. A server is available to add to the cluster only if it is not a member of another cluster.

   If no servers are available to add to the cluster, you can create a new server by entering the following values, then selecting Add New Server:

   - Server Name – a name for the new server.
   - Host Name – the name of the machine that hosts the new server. If the name of the server is different than the name of the machine, enter the fully-qualified machine name; for example, `mach1.sybase.com`.
   - IIOP Port Number – the server's IIOP port number.
   - HTTP Port Number – the server's HTTP port number.

   A new server is created with an IIOP listener and an HTTP listener. The listeners are named *serverName*_iiop and *serverName*_http, where *serverName* is the name of the new server. After you create a new server, the server displays in the Participating Servers list, and it is selected as a member of the cluster.

   To add another server, repeat this step.
5. On the Summary page, verify the cluster configuration. Select Back to return to a previous page, or select Finish to create the cluster.

   When you create a cluster, these export configuration files are generated automatically:

   - sync-*clusterName*, which you can use to synchronize the cluster. *clusterName* represents the name of the cluster.

- sync-*clusterName*-security, which you can use to synchronize security changes, such as new users, roles, and role assignments.

See "Creating export configurations" on page 106.

Cluster members must all have the same administrative password All servers in a cluster must have the same password for the admin@system user. This password is used to establish a trust relationship between the servers. Load balancing relies on trusted interserver communication, as does JMS clustering.

❖ Adding servers to a cluster

If all the servers in a cluster run on separate machines, all the servers can use the same listener ports. If multiple servers in a cluster run on the same machine, they must use different listener ports.

Use the Management Console to connect to the primary server, and add participating servers to the cluster.

1. Expand the Clusters icon.
2. Highlight the cluster to which you want to add servers.
3. On the General tab, enter the following information, then click Apply:
   - **Primary Server** Select the primary server in the cluster; the default is the server to which you are connected. The configuration property name is clusterPrimary.
   - **Cluster ID** A unique ID for the cluster. The typical format is a dotted-decimal TCP/IP address, a colon, and a port number; for example, 10.11.12.13:2000. The ID must be unique across the network of connected servers that share the same database. The ID need not be globally unique or refer to an actual TCP/IP address. The default value is derived from either the iiopListeners property or the httpListeners property of the server that is first added to the cluster. You can specify the value explicitly. The configuration property name is clusterID.
   - **Cluster Version** The version number of the cluster. The configuration property name is clusterVersion.

- **Server Properties Not to Synchronize** A comma-separated list of server properties that are not propagated during cluster synchronization; by default, httpListeners, iiopListeners, host.name, and processID. The configuration property name is serverPropertiesNotToSynchronize.
- **Participating Servers** To add all the defined servers to the cluster, select All. To add individual servers to the cluster, choose Select, then choose each server. If you select None, the cluster contains only the primary server. The configuration property name is servers.

4 After you add servers to the cluster, configure and build the cluster's export configuration to enable synchronization—see "Defining the components of an export configuration" on page 107. When you configure the export configuration, select Restart Servers After Synchronizing to ensure that all the servers in the cluster use the same security-key information for interserver configuration.

If you add more servers later, you must synchronize the cluster again.

Warning! After you add a nonprimary server to a cluster, the Management Console warns you when you connect directly to that server. If the new server has been the target of at least one synchronization before it was added as a cluster member, direct user updates to the server's configuration may be overwritten when the cluster is synchronized.

Sample cluster

This example illustrates how to create a new server and a cluster that contains the default server and the new server.

❖ **Creating a new server and a cluster**

1 While connected to the default server, create a new server:

a In the Management Console, highlight Servers, right-click, and select Add.

b Enter the following, then select Next:

- Server Name – the name of the server.
- Server Host Name – the name of the machine on which the server runs; for example, `machine2.sybase.com`.

  If the server and the machine on which it runs have the same name, you need not enter the domain.

   c. Add an IIOP listener by selecting Create a New Listener, clicking Add Listener, then entering:
      - Listener Name – a name for the IIOP listener.
      - Listener URL – select IIOP, then enter the machine name and the IIOP port number.
   d. Add an HTTP listener by clicking Add Listener, then entering:
      - Listener Name – a name for the HTTP listener.
      - Listener URL – select HTTP, then enter the machine name and the HTTP port number.
   e. Select Finish.
2. Create a cluster that includes the default server and the new server:
   a. Select the Clusters node, right-click, and choose Add.
   b. In the wizard, enter a name for the cluster, and select Finish.
   c. In the left pane, highlight the cluster, then on the General tab:
      - Primary Server – select the server to which you are connected.
      - Participating Servers – choose Select, then choose the server you just created.
   d. Click Apply.

❖ Removing a server from a cluster

1. Expand the Clusters icon, and select the cluster from which to remove the server.
2. On the General tab, unselect the server from the list of participating servers. To remove all the servers from the cluster, select None.
3. Click Apply.
4. Re-create the cluster's export configuration, and synchronize the cluster—select Restart Servers After Synchronizing to restart the participating servers.

❖ Deleting an existing cluster

1. Expand the Clusters icon, and select the cluster to delete.
2. Right-click, and select Delete. In the Delete wizard, confirm that you want to delete the cluster.

Cluster partitions

A cluster partition allows you to define a subset of a cluster consisting of primary and backup servers. You can configure individual components, message queues, and topics to be serviced only by the servers in a specific partition. Each server can be assigned to multiple partitions.

A cluster can contain one or more partitions, and each partition identifies preferred servers and alternate servers. Cluster partitions provide an effective way to perform load balancing without sacrificing high availability. Client requests are handled by one or more of the preferred servers in a partition, unless there are none available, in which case, the alternate servers handle the requests.

You can implement “hot standby” by creating a cluster partition with two members, a preferred server and an alternate server.

To facilitate cluster synchronization, Sybase recommends that you do not assign a cluster’s primary server to a cluster partition.

❖ **Creating a cluster partition**

1 In the Management Console, highlight the Cluster Partitions icon, right-click, and select Add.

2 Complete the wizard, and select Finish.

3 Add servers to the cluster partition.

❖ **Configuring a cluster partition**

1 Expand the Cluster Partitions folder, and select the cluster partition to configure.

2 On the General tab, enter:

- **Load Balancing Monitor** Select the load balancing monitor from the list. The configuration property name is loadBalancingMonitor.

  Each cluster partition can use a different load-balancing monitor, however, it is best to choose a load-balancing monitor that matches the thread monitor being used by the servers across which the load is balanced. To determine which thread monitor is being used, check the web.threadMonitor, ejb.localThreadMonitor, and ejb.remoteThreadMonitor properties in the Ant configuration scripts for your deployed EJB and Web modules.

- **Resolve Cache Timeout** The number of seconds for a server to wait before rereading the property files to detect cluster changes (new or deleted cluster members); the default is 60 seconds. The configuration property name is resolveCacheTimeout.
- **Preferred Servers** Select the preferred servers for this partition. Choose Select, then pick the servers individually. To add all the servers, select All; to add none of the servers, select None. The configuration property name is preferredServers, which defines the preferred servers in a comma-separated list.
- **Alternate Servers** Select the alternate servers for this partition. If the preferred servers are unavailable, the alternate servers handle the requests. To add all the servers, select All; to add none of the servers, select None. The configuration property name is alternateServers, which defines the alternate servers in a comma-separated list.

❖ Assigning an EJB component to a cluster partition using the Management Console

1. In the Management Console, expand the EJB Modules folder, and select the EJB to configure.
2. On the Advanced tab, enter the name of the cluster partition, then click Apply.
3. Select Synchronize to update the value of the cluster partition in the Ant configuration file that defines the EJB.

❖ Assigning a message queue or topic to a cluster partition using the Management Console

1. In the Management Console, expand the Resources folder and either the JMS Message Queues or JMS Message Topics folder.
2. Select the message queue or topic to configure.
3. On the General tab, select the cluster partition from the list. Click Apply.

❖ Assigning an EJB component to a cluster partition by editing the Ant configuration file

- In the Ant configuration file that defines the EJB component, enter the following property definition, where *myPartition* is the name of the cluster partition:

```
<clusterPartition partition_name = “myPartition”/>
```

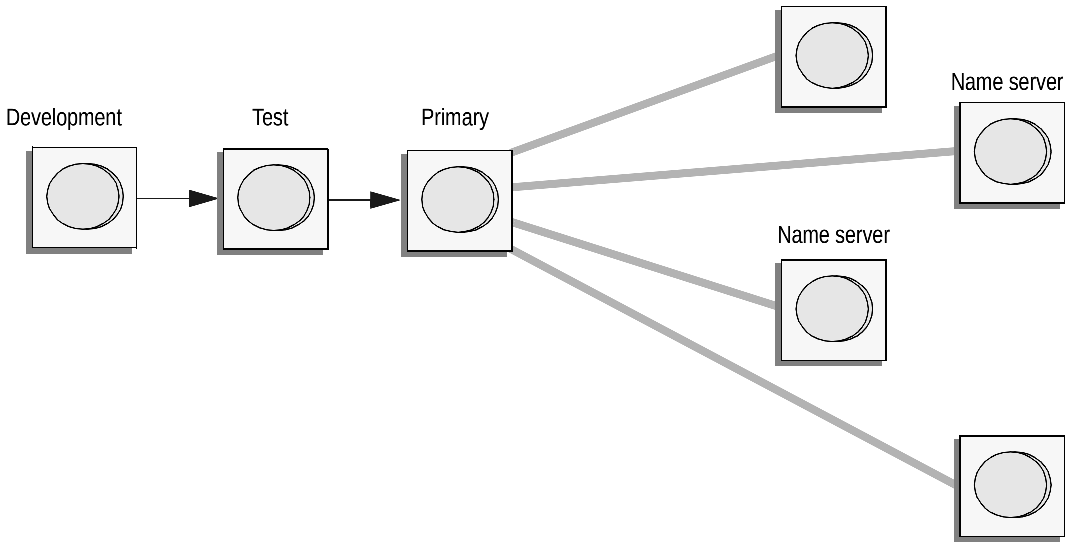
Synchronization

Synchronization provides an easy way to distribute modules and configuration information between servers. You can 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 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.

For example, as Figure 7-2 illustrates, you can replicate new components from a testing or development server to one or more production servers.

Figure 6-1: Synchronization example



If you are using clusters and make configuration changes to the primary server, the “admin@system” user (or any user who is a member of the admin-role role) can synchronize those changes to participating nonprimary servers.

To synchronize a cluster, you must be connected to the primary server of 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.

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 the Management Console or jagtool and delete the entity manually. However, even if you do not delete the entity, resynchronizing the cluster removes the entity from the list of modules that are started automatically, for each target server.

You can synchronize a single server or a cluster of servers.

Shut down services before synchronizing Before you synchronize a cluster, shut down any services that are running. If you do not, some files required for synchronization may be locked. To prevent ordinary clients from connecting to the primary server, and to make it easier to shut down services, do not assign the primary server to a cluster partition.

To synchronize a server or cluster, create an export configuration to define the components to synchronize—see Chapter 7, “Exporting Server Modules.”

CHAPTER 7 Exporting Server Modules

This chapter describes how to create customized modules called export configurations to export parts of a server for deployment, synchronization, or backup.

Overview

Using an export configuration, you can export an entity, such as a Web application, or specify files to be synchronized to another server or to a cluster of servers. You can create an export configuration that contains specific deployment options for a server; for example, a server without a Web container, or a server that contains only an EJB container. You can also use export configurations to create a backup of a server or parts of a server.

An export configuration includes a set of EAServer files, Java classes, and entities, which are built into a *.zip* file. Entities include:

- Application clients
- Applications
- Connectors modules
- EJB modules
- Web applications

Export configurations can also include information about cluster synchronization and dependencies to other export configurations.

You can define an export configuration that includes specific parts (functionality) of a server, which can be reused by other export configurations. For example, you might create one export configuration that defines a Web server and another that defines an EJB server. To create a Web server and an EJB server, a third export configuration could specify the first two export configurations as dependencies.

You can add classes in an export configuration by specifying either complete directories, or individual classes, including those within JAR files.

If you include an entity in an export configuration, all the files necessary for the entity are exported, including configuration files, property files, and classes.

You can add any file on the server's file system to an export configuration. Files are stored in the export configuration *.zip* file in the same location where they exist in the current file system.

An export configuration can include dependencies to one or more other export configurations, which may contain classes, entities, and files.

Creating export configurations

You can create an export configuration either using the Management Console or on the command line. After you create an export configuration, define the classes and entities to include, and if appropriate, define the dependencies. Finally, build the export configuration to make it available for synchronization.

When you create a cluster, these export configuration files are generated automatically:

- sync-*clusterName*, which you can use to synchronize the cluster. *clusterName* represents the name of the cluster.
- sync-*clusterName*-security, which you can use to synchronize security changes only.

❖ Creating an export configuration using the Management Console

1 In the left pane, highlight Export Configurations, right-click, and select Add.

2. Complete the wizard to create a new export configuration.
3. Define the components of the export configuration.

❖ **Creating an export configuration on the command line**

- Change to the *bin* subdirectory of your EAServer installation, and run:

```
export [-exportDir path] [-exportGenFiles] [-server host:port] entities
```

See export on page 181.

❖ **Defining the components of an export configuration**

1. In the left pane, select the export configuration to define.
2. In the right pane, define the export configuration properties on the tabs described below. Select Apply on each tab to save your changes.
3. To use the export configuration for synchronizing or backup, build the export configuration—see “Building an export configuration using the Management Console” on page 112.

General tab

The General tab displays the following properties; you can edit Description and Configuration Type, all other properties are read-only.

- **Last Build Time** The most-recent time the export configuration was built. The configuration property name is lastBuildTime.
- **Download URL** The URL from which you can download the export configuration *.zip* file. The configuration property name is downloadURL.
- **XML Config File** An XML configuration file to include in the export configuration. The file defines tasks to perform after the export configuration has been applied or synchronized to a server. The configuration property name is xml-config.
- **EAS Display Name** The name of the export configuration as it displays in the Management Console. The configuration property name is display-name.
- **Description** A description of the export configuration. The configuration property name is description.
- **Configuration Type** Select one of:

- Abstract – you can use an abstract export configuration as the foundation for other, buildable export configurations. You cannot build an abstract-type export configuration and use it for synchronization.
- Patch – a set of files to apply to another server.
- Server – select this type if the export configuration contains an entire server.

The configuration property name is configType.

Class Files tab

On the Class Files tab, define the class files to export:

- **Source Class Path** Enter a comma-separated list of paths to directories or JAR files from which you can select classes to export. The configuration property name is jarFileRoot.
- **Included Classes** Select the class files you want to export. If a class file exists in more than one location in the source class path, the class file that is closer to the beginning of the source class path is the one that is added to the export configuration. The configuration property name is jarFileList.

  To remove a class from the list, click Remove next to the name of the class.
- **Create Runtime JAR** Select to create a new runtime JAR file that contains only the classes you choose. The name of the JAR is based on the current JDK runtime version. By default, JDK 1.4 is the runtime, and the name of the JAR is *eas-server-14.jar*. If you do not select this option, the JAR contains all the EAServer runtime classes. A JAR file with fewer classes improves performance. The configuration property name is createRuntimeClassesJar.

Dependencies tab

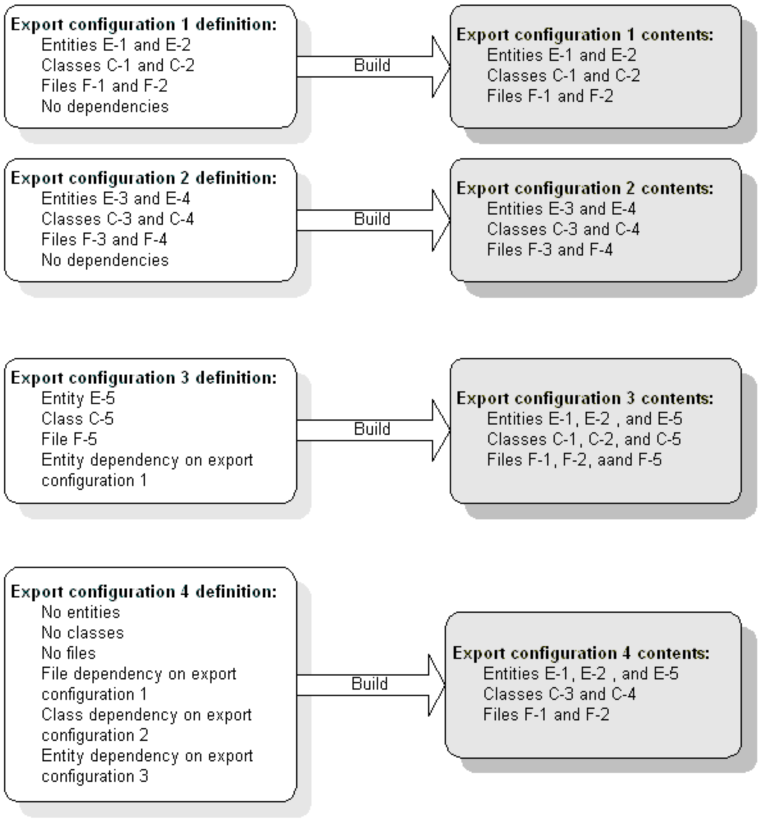
On the Dependencies tab, you can select export configurations that contain entities, classes, or files on which this export configuration depends.

Dependencies provide a way to group classes, entities, and files into logical units, which you can include in other export configurations. For example, you could create an export configuration *ec1* that includes two Web applications. To include the same two Web applications in another export configuration, specify *ec1* as an entity dependency.

Figure 7-1 on page 110 illustrates how dependencies affect the contents of an export configuration after it is built. For example, *export configuration 4* has an entity dependency on *export configuration 3*, which in turn has an entity dependency on *export configuration 1*. Therefore, when *export configuration 4* is built, it includes the entities in *export configuration 3* and *export configuration 1*.

- **Class Dependencies** Select one or more export configurations that contain classes on which the current export configuration depends. To specify that all the export configurations contain classes on which the current export configuration depends, select All; if none of the export configurations contain classes on which the current export configuration depends, select None. The configuration property name is classDependencies.
- **Entity Dependencies** Select one or more export configurations that contain entities on which this export configuration depends. As described for Class Dependencies, you can also select All or None. The configuration property name is entityDependencies.
- **File Dependencies** Select one or more export configurations that contain files on which the current export configuration depends. As described for Class Dependencies, you can also select All or None. The configuration property name is fileDependencies.

Figure 7-1: Export configuration dependencies


Export configuration 1 definition:
Entities E-1 and E-2
Classes C-1 and C-2
Files F-1 and F-2
No dependencies
Build
Export configuration 1 contents:
Entities E-1 and E-2
Classes C-1 and C-2
Files F-1 and F-2
Export configuration 2 definition:
Entities E-3 and E-4
Classes C-3 and C-4
Files F-3 and F-4
No dependencies
Build
Export configuration 2 contents:
Entities E-3 and E-4
Classes C-3 and C-4
Files F-3 and F-4
Export configuration 3 definition:
Entity E-5
Class C-5
File F-5
Entity dependency on export configuration 1
Build
Export configuration 3 contents:
Entities E-1, E-2 , and E-5
Classes C-1, C-2, and C-5
Files F-1, F-2, aand F-5
Export configuration 4 definition:
No entities
No classes
No files
File dependency on export configuration 1
Class dependency on export configuration 2
Entity dependency on export configuration 3
Build
Export configuration 4 contents:
Entities E-1, E-2 , and E-5
Classes C-3 and C-4
Files F-1 and F-2

Entities tab

On the Entities tab, you can include application clients, applications, connectors, EJBs, and Web applications in the export configuration.

If you include an entity in an export configuration, all the files associated with said entity are exported. For example, if you export a Web application, the files in the *deploy*, *config*, and *Repository* directories that are required for the Web application are automatically added to the export configuration.

- **Include Entities** Select the entities to include in the export configuration file. The configuration property name is entitiesList.

To remove an entity from the list, click Remove next to the name of the entity.

- **Include Generated Files** Select to include files that were generated by the server. The configuration property name is exportEntityGenFiles.

Other Files tab

On the Other Files tab, you can add files other than class and entity files to the export configuration.

- **Include Other Files** Select the files to include. You can select individual files or entire directories. After you select a file or directory, it displays as an item in the list of included files. You may need to scroll to see the entire list. The configuration property name is includeOtherFiles.

  You can specify a set of files using a wildcard character; for example, to include all the property files in a directory, select the directory, then in the list of included files, enter: `/*.properties` next to the directory name.

  To copy a server's repository to another server or cluster, select the *Repository* directory tree. When you synchronize, the files are applied on top of the destination server's repository.

  You cannot export a server's **.properties* file; for example, if the name of the server is "fury," you cannot export the *fury.properties* file, which is located in *\Repository\Instance\com\sybase\djc\server\ApplicationServer*.

  To remove an item from the list, select Remove next to the file or directory item.

Synchronization tab

On the Synchronization tab, define the properties required to use the export configuration to synchronize one server to another server or to all the servers in a cluster:

- **Synchronize Server** Select the servers to synchronize. To synchronize all the servers, select All. If you do not want to synchronize any of the servers, select None. The configuration property name is synchronizeTo, and it defines the list of both servers and clusters to synchronize.
- **Synchronize Cluster** To synchronize the contents of the export configuration with the servers in one or more clusters, select the clusters. To synchronize the servers in all the clusters, select All. If you do not want to synchronize the servers in any of the clusters, select None. The configuration property name is synchronizeTo.
- **Build Before Synchronizing** Select this option to build the export configuration before synchronizing another server or cluster. By default, this option is enabled. The configuration property name is buildBeforeSync.
- **Update After Restart** If you select this option, an export configuration *.zip* file, located in the *%DJC_HOME%\temp\sync* directory on the destination server, is unzipped when the server is restarted. This prevents "file in use" errors that may occur while updating files that are in use by a running server. The configuration property name is updateAfterRestart.
- **Restart Servers After Synchronizing** Select to restart the destination servers after unzipping the export configuration on each server. The configuration property name is restartAfterUpdate.
- **Restart Timeout** Specify the number of seconds to wait before abandoning attempts to restart a server. The default is 300 (5 minutes). The configuration property name is restartTimeout.

❖ Building an export configuration using the Management Console

- Highlight the export configuration, right-click, and select Build.

A JAR file is created, which contains the classes, entities, and files that are directly included in the export configuration, and its dependencies. Then the JAR file and a *.properties* file, which defines the JAR, are compressed into a *.zip* file.

❖ Synchronizing an export configuration using the Management Console

1. Highlight the export configuration, right-click, and select Synchronize.
2. In the wizard, specify the server to synchronize.

The export configuration *.zip* file is copied to the */temp/sync* subdirectory on the remote server, and the server is restarted.

Note The user name and password must be the same on all the servers that you want to synchronize.

❖ **Building or synchronizing an export configuration on the command line**

- Change to the EAServer *bin* directory, and run:

  ```
  export-config -action action [-verbose] name
  ```

  where *action* is the action to perform (build or synchronize) and *name* is the name of an existing export configuration—see export-config on page 182.

CHAPTER 8

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 use Java Connection Management (JCM) with EAServer database connectivity to implement the Adaptive Server Enterprise failover feature.

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

EAServer uses a three-step process to perform load balancing:

1. Randomly choose up to three servers in a cluster partition; fewer if the cluster partition has fewer than three servers.
2. From each server, get load metrics, which is the value of waitingThreadCount from the loadBalancingMonitor. This requires interserver communication.
3. Organize the interoperable object reference (IOR) profiles according to the load metrics, least loaded first (lowest value of waitingThreadCount).

Interoperable object references

Load balancing uses the EAServer naming service to distribute incoming IIOP requests across the servers within an EAServer cluster partition:

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.

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 cluster partition has at least two member servers.

To guarantee end-to-end high availability, clients should use URLs of the form "iiop://host1:9000;iiop://host2:9000;..." when performing any of the activities listed below. This ensures that no part of a client's initialization is limited to a single point of failure. 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.

In C++ or PowerBuilder clients, the host name used in the initial URLs should not match the host name in the server's listener definitions. To achieve this, either use IP addresses for listeners and host names in clients, or map multiple host names to the same IP address. This prevents your name server from performing the work of the member servers. For example, if the client's IIOP connection pool decides that since it has already connected to a name server that is also a member server hosting a component that the client wants, the client does not need a connection to the preferred member server. In Java clients that use the EAServer 6.0 client runtime, there are separate connection pools for naming and non-naming purposes, so this issue is avoided.

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.

Setting up and using the Web-load balancer

The Web-load balancer is a Web application that balances server loads by redirecting HTTP requests from Web applications to application servers within a cluster. In some cases, the Web-load balancer provides a simpler alternative to the Web redirector.

Configuring the Web-load balancer

All servers in your cluster must have at least one IIOP listener defined, since the Web-load balancer uses IIOP for interserver communication. To configure the Web-load balancer:

1. In the Management Console, expand the Servers folder, and select the server.
2. On the Start Modules tab, under System Start Modules, verify that *webapp-wlb* is enabled; it is by default. If you change any values, click Apply.
3. Define a cluster, and optionally, define one or more cluster partitions.
4. Add servers to the cluster, and optionally, to the cluster partitions.
5. Verify that all the servers in the cluster have at least one IIOP listener defined.

Using the Web-load balancer

To use the Web-load balancer:

1. Define a cluster using either the Management Console or an Ant configuration script. You may optionally define one or more cluster partitions.
2. To force load balancing of a URL, such as http://my.example.com/MyApp/MyPage, include a load balancing context path prefix, such as “~/MyCluster” or “~/MyPartition”; for example:

   ```
   http://my.example.com/~/MyCluster/MyApp/MyPage
   ```

   The tilde (“~”) prefix is the default context path for the *wlb* Web application. To change this, edit the *config/webapp-wlb-user.xml* configuration file, run its configure target, then restart the server.

3 Verify that the server name specified in the load balancing URL resolves to, or is routed to, either a member of the cluster, or a server with the same administrative password. The server name specified in the URL need not resolve to the same server every time; for example, you can use a round-robin distribution policy.

The Web-load balancer selects one active server in the cluster (or partition). A "weighted-random" algorithm is used. Three servers (or fewer, if three are not available) are randomly selected from the cluster (or partition) and the least loaded of the selected servers is the target of the HTTP redirect, with the load-balancing context path prefix removed from the URL. An example of a resolved URL is:

http://server27.example.com/MyApp/MyPage

Where "http://server27.example.com" is the primary HTTP listener for the selected application server. If your application server has multiple HTTP listeners, a port number is selected if it matches the port number (modulo 10) of the initial load-balancing URL. For example, if a server has two HTTP listeners:

- http://server27.example.com:80
- https://server27.example.com:443

the load-balancing URL "http://my.example.com/~/MyCluster/MyApp/MyPage" might resolve to "http://server27.example.com:80/MyApp/MyPage." Whereas the load-balancing URL "https://my.example.com:443/~/MyCluster/MyApp/MyPage" might resolve to "https://server27.example.com:443/MyApp/MyPage." If unspecified, the port number in a URL defaults to 80 for HTTP and 443 for HTTPS.

Once a redirect is performed, the load balancer is not involved in subsequent requests, unless the Web application at the target URL delivers pages that include a load-balancing context path prefix.

If the target Web application is not distributable, and uses HTTP sessions, the Web-load balancer should be used only before a Web session is created. Otherwise, an HTTP redirect from the server might occur, in which a Web session is established to a server that does not exist in the Web session.

Note HTML pages delivered by the target Web application should use relative paths (without the http://my.example.com prefix). If absolute paths are used in links, the server selected by the Web-load balancer is not used for a link with an absolute path, unless the absolute path includes a load balancing context path prefix. Frequent use of the load balancer degrades performance, compared with using it only when a session is established.

The server name used in a load-balancing URL (for example, my.example.com) could become a single point of failure in your system, unless you use an external mechanism (hardware redirector or DNS round robin) to ensure that initial (preload balancing) requests are protected from a single point of failure.

Tuning the Web-load balancer

The weighted-load distribution policy uses the current number of waiting threads for the selected thread monitor to determine the least-loaded server.

If the load-balancing context path prefix uses:

- A cluster name, the default thread monitor is used
- A partition name, the default thread monitor is used, but you can change this by setting the loadBlancingMonitor property for the partition

Load balancing works best when the value of the thread monitor’s maximumActiveThreads property is nonzero. If the value is zero, there is no limit.

As a general rule, set the value of maximumActiveThreads to 20 threads per CPU. Sybase recommends that you test your system's performance to determine whether this value is optimal.

Note A cluster may include any number of (possibly overlapping) partitions, and each partition may use a different load-balancing monitor. However, it is generally only appropriate to choose a load-balancing monitor that matches a thread monitor in use by the servers across which the load will be balanced. Check the web.threadMonitor, ejb.localThreadMonitor and ejb.remoteThreadMonitor properties in the Ant configuration scripts for your deployed EJB and Web modules, to determine which thread monitors are appropriate for load balancing.

The default thread monitor is preconfigured with a maximumActiveThreads property value of zero. Change this to a nonzero value to use the Web-load balancer with a load-weighted random distribution.

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 Chapter 7, "Exporting Server Modules." The sections below describe additional configuration that is required for components deployed in a cluster.

Partitioned components

Partitioning restricts components to a subset of servers within a cluster. A cluster partition allows you to define a subset of a cluster consisting of primary and backup servers. You can configure individual components to be serviced only by the servers in a specific partition. See "Cluster partitions" on page 101.

Note You may want to use partitioning to separate CPU-bound components from database-bound components.

Automatic failover for components

You can 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 provides transparent failover, unless all the servers in the cluster fail.

Automatic failover is not the default for EAServer components. If a component does not support automatic failover, and the server hosting that component fails, a client can still fail over if either the client's automaticFailover initial context property is enabled, or if failover is enabled for the EJB provider.

❖ **Setting automatic failover for a component**

- In the component's Ant configuration file, set the automaticFailover subtask of the setProperties task to true.

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. If you use stateful session beans that are enabled for automatic failover, this is not an issue.

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.

❖ Configuring EJB session beans to ignore duplicate calls during failover

- To configure EJB session beans to ignore duplicate calls during failover, use request and response logging:
  - For void business methods, enable request logging by setting this property in the EJB module's Ant configuration file:

    ```
    <automaticFailover enableRequestLog=”true”/>
    ```

  - For non-void business methods, enable response logging by setting this property in the EJB module's Ant configuration file:

    ```
    <automaticFailover enableResponseLog=”true”/>
    ```

Logging uses a database. Log records are automatically deleted unless automatic failover occurs, in which case the log records are retained to enable duplicate detection.

❖ Programmatically avoiding duplicate inserts during failover

To help avoid duplicate inserts during failover, you can add a component method 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 *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

One way to distribute a Web application is by deploying it to an EAServer cluster. A distributed Web application can provide better performance since multiple machines can handle more load than one machine. Clusters also provide high availability; if one machine goes offline, clients can connect to another server in the cluster.

Alternately, you can support failover and load balancing using a Web server redirector plug-in—see the *EAServer Installation Guide*.

Web applications manage state information via HTTP sessions. Distributed HTTP sessions are required to support failover—see “Tuning distributed HTTP session settings,” in Chapter 5, “Web Application Tuning,” in the *EAServer Performance and Tuning Guide*.

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

Beginning in EAServer 6.0, the *wlb* Web application (Web-load balancer) provides load balancing for Web applications using HTTP redirects—see “Setting up and using the Web-load balancer” on page 118.

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 persistent storage. EAServer stores session data in a persistent data store to support shared sessions and session failover.

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.

Because 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 persistent session storage

EAServer stores all session data in a remote database, connecting through the predefined JDBC data source *session.db*. All servers in the cluster share the same database. Sybase recommends that you configure this data source 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 database table that stores session data is automatically created for every database type that EAServer supports. The name of the table is web_session.

Change the sample session.db properties As preconfigured, the *session.db* data source 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

The Web application must be marked "distributable."

Perform these steps from the Management Console, while connected to the primary server for your EAServer cluster:

1. Install the Web application to one or more logical servers that are part of the cluster.
2. Configure the properties of the data source named *session.db* to connect to the database that you use for persistent session storage. This data source must use JDBC. See "Configuring data sources" on page 66 for instructions.
3. Make sure the *session.db* data source is installed in each logical server where the Web application is installed.
4. Synchronize the EAServer cluster to propagate the configuration changes to other servers in the cluster. See Chapter 7, "Exporting Server Modules."

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 data sources or the Sybase Open Client Client-Library.

Data sources

Configure data sources by enabling Java Database Connectivity (JDBC) connections to establish failover-enabled connections to Adaptive Server Enterprise. See the jConnect and JNDI documentation for more details on the jConnect configuration.

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 JNDI with JDBC 1.0 and JDBC 2.0 connections by configuring a data source.

Use the Management Console to configure the data source with these values for the specified properties:

- User – the database user name.
- Password – the database password.
- Database URL – `jdbc:sybase:Tds:${`***`serverName`***`}:${`***`portNumber`***`}`.

JDBC 2.0 extension/JTA drivers

Set up JDBC 2.0 extension / JTA drivers by 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, "CORBA Component Life Cycles and Transaction Semantics," in the *EAServer CORBA Components Guide*.

Using the Management Console:

1. Configure the data source with these values for the specified properties:
   - User – the database user name.
   - Password – the database password.
   - Database URL – `jdbc:sybase:Tds:${`***`serverName`***`}:${`***`portNumber`***`}`.
2. Set the JDBC Driver Class property for the database type that is configured for your data source to com.sybase.jdbc2.jdbc.SybXADataSource.

Troubleshooting the database connection

To find out if your database connection is working, ping the data source.

❖ **Pinging the data source**

- In the Management Console, select the data source, right-click, 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 *providerutil.jar*, *jconn2.jar* (the jConnect driver), and *jndi.jar* (JNDI 1.2)
- You are using JDK 1.4 or JDK 1.5.

Open Client Client-Library

You can establish high-availability Client-Library connections to an Adaptive Server Enterprise database by:

- Modifying the client connection information
- 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*%DJC_HOME%\ini\sql.ini* file:

```
[Server1]
master=NLWNSCK,Server1-host,5200
query=NLWNSCK,Server1-host,5200
hafailover=Server2

[Server2]
master=NLWNSCK,Server2-host,5300
query=NLWNSCK,Server2-host,5300
hafailover=Server1
```

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://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_15.0.ha_avail/html/ha_avail/title.htm.

C H A P T E R 9

Importing Application Components

This chapter describes how to deploy application components to EAServer using standard J2EE archive formats and EAServer JAR format, including the import of archive files for:

- EJB modules
- CORBA packages
- Web applications
- J2EE applications
- J2EE application clients
- J2EE connectors

To export application components, see Chapter 7, “Exporting Server Modules.”

Deploying packages and components

Components must be archived as part of the package where they are installed. You can import two archive 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. See Chapter 2, "Deploying and Configuring EJB Components," in the *EAServer Enterprise JavaBeans User's Guide*.

You cannot deploy EJB components and components of other types (such as PowerBuilder or CORBA) in the same package.

❖ **Deploying packages between servers using an export configuration**

1. Start the Management Console and connect to the test server.
2. Create an export configuration that includes the files you want to import to your production server—see Chapter 7, "Exporting Server Modules."

   When you create the export configuration, select the following options on the Synchronization tab:

   - Build Before Synchronizing
   - Synchronize Server or Synchronize Cluster
   - Restart Servers After Synchronizing

   By default, all the files that are required for an entity are included in the export configuration. You can include other files, such as project descriptions, client applets, and HTML pages, when you configure the export configuration.
3. Start the Management Console on the production server host, and connect to the production server.
4. If you have copied a package between host machines that have different architectures, recompile the components to run on the new architecture.

Importing components in EAServer JAR format

The Management Console allows you to import an archive file containing component definitions and implementation files for all components in a package or a single component. 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.

❖ **Importing a package archive**

1. Copy the JAR file containing the package definition to the host machine for the target server.
2. Start the Management Console and connect to the target server.
3. Expand the CORBA 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, right-click, and select Deploy.
5. In the Deploy wizard:
   a. Use the Browse button to locate the JAR file that you are importing.
   b. 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 Finish.

❖ **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. Right-click, and select Deploy.
3. Use the Browse button to select the EAServer JAR file, or type the full path to the file.
4. Complete the wizard to deploy the component.

Importing 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 JBuilder to define and develop beans and create an EJB-JAR file. You can import JAR files in the EJB 1.1, EJB 2.0, or EJB 2.1 formats. EAServer reads the JAR file and creates a package containing a component for each bean in the JAR file.

Home names for imported EJB components EAServer sets an imported bean's home name to the EAServer default, *package/component*, where *package* is the Management Console package name, and *component* is the component name.

❖ Importing an EJB-JAR file

1. Start the Management Console if it is not already running, and connect to the server where you want to install the component.
2. Highlight the top-level EJB Modules folder, right-click, and select Deploy.
3. The Deploy wizard guides you through the process of deploying an EJB-JAR file.

   EAServer creates a new package that contains a component for each bean defined in the JAR file. The new package name is "ejbjar-" followed by the JAR name; for example, if you deploy an EJB-JAR called "helloworld.jar," the package name is "ejbjar-helloworld." For each interface in the EJB-JAR—remote home, remote interface, local home, local interface, and service endpoint interface—EAServer creates one lightweight component; thus, mapping of EJBs to components is one-to-many.
4. Click Finish.

See Chapter 2, "Deploying and Configuring EJB Components," in the *EAServer Enterprise JavaBeans User's Guide*.

❖ Exporting an EJB module

- To export an EJB module, create an export configuration that includes the EJB module—see Chapter 7, "Exporting Server Modules."

Deploying Web applications

Using the Management Console, you can import Web applications into EAServer or export Web applications from EAServer 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. See "Deploying Web applications" in Chapter 1, "Defining Web Applications," in the *EAServer Web Application Programmer's 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.

❖ **Importing a Web application**

1 Highlight the top-level Web Applications folder, right-click, and select Deploy.

  The Deploy wizard guides you through the process of deploying a WAR or EAServer JAR file.

2 In the Deploy wizard:

  a Use the Browse button to locate the archive file that you are importing.

  b 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.

3 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.

❖ **Exporting a Web application**

- To export a Web application, create an export configuration that includes the Web application—see Chapter 7, "Exporting Server Modules."

What is created during import

When importing an 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 named "webapp-" followed by the WAR name; for example, if you deploy an WAR called "samplewebapp.war," the Web application name is "webapp-samplewebapp." 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 Web application settings:

- Role mappings
- Resource references
- EJB references
- Environment properties

Deploying J2EE applications

Using the Management Console, you can import applications into EAServer, or export applications from EAServer 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 142.

- **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.

❖ **Importing an application**

1. Highlight the top-level Applications folder, right-click, and select Deploy.
2. In the Deploy wizard:
   a. Enter the name of the archive file, or click Browse and select the file.
   b. Enter a name for the application module, and specify whether to overwrite an existing module with the same name.
   c. Specify whether to perform validation of the deployment descriptor. EAServer performs validation by default in accord with the EJB specification. However, disabling validation may allow you to deploy archives that have invalid XML but are otherwise correctly packaged.
   d. Select the server in which to install the module, or select not to install the module in a server.
   e. If the archive file does not include an EAServer configuration file in the *META-INF* directory, the system generates one automatically. To save a copy of the archive with the automatically-generated configuration file, enter the location.
   f. Select Finish to deploy the application.

❖ **Exporting an application**

- To export an application, create an export configuration that includes the application—see Chapter 7, “Exporting Server Modules.”

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 EAR file.
- For each EJB-JAR file in the EAR, an entity called “ejbjar-” followed by the JAR name; for example, if the application contains an EJB-JAR called “helloworld.jar,” the entity name is “ejbjar-helloworld.”
- 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, *module/component*, where *module* is the Management Console EJB module name, and *component* is the Management Console component name.

- For each WAR file in the EAR, a Web application named “webapp-” followed by the WAR name; for example, if the EAR contains a WAR called “samplewebapp.war,” the Web application name is “webapp-samplewebapp.”
- 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 an EAR file does not contain EAServer XML configuration files, you may need to configure the following component or Web application properties 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 are configured by the Deploy wizard.

Use the status window as a to-do list In the deployment status window, the Management Console 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.

J2EE application configuration

The Management Console displays the contents of a deployed application on the following tabs. If you update any values, click Apply to save your changes:

- **General** On the General tab, you can enter a description for the application, and specify the class loader.
- **AppClientModule** Contains links to the application client modules in the application. To display the configuration properties for an application client, click its link.
- **application.xml** Displays the contents of the deployment descriptor.
- **Configuration** Displays the Ant configuration script that defines the application. To add configuration properties or change existing property values, see “Configuring application properties.”
- **User Configuration** Displays the application’s user configuration file. To add configuration properties or change existing property values, see “Configuring application properties.”
- **EJB Modules** Contains links to the EJB modules in the application. To display the configuration properties for a EJB module, click its link.
- **Web Modules** Contains links to the Web modules in the application. To display the configuration properties for a Web module, click its link.

❖ Configuring application properties

To update application properties, edit the user configuration file, rather than the Ant configuration script.

1. If you see a User Configuration tab, select it; otherwise:
   a. In the left pane, highlight the application, right-click, and select Create User Configuration.
   b. Highlight the application again, right-click, and select Refresh Node. A user configuration script is created and displays in a new User Configuration tab. Select the tab.
2. Add properties and values you want to change. You can copy property definitions from the Ant configuration script, then edit the values. If you reconfigure or recompile the application, the property values in the user configuration script override the property values in the Ant script.

   For information about creating and configuring Ant scripts and user configuration scripts, see Chapter 2, “Ant-Based Configuration,” in the *EAServer Automated Configuration Guide*.

3 Recompile the application. In the left pane, highlight the application, right-click, and select Run Ant Recompile.

4 Refresh the server.

Deploying application clients

Using the Management Console, you can export application clients from EAServer 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 about creating, configuring, and running application clients, see Chapter 4, "Creating Application Clients," in the *EAServer Enterprise JavaBeans User's Guide*.

❖ Deploying application clients

1. Highlight the Application Clients folder, right-click, and select Deploy.
2. The Deploy wizard guides you through the process of deploying the application client.
3. The Deploy wizard reads the archive 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

- To export an application client, create an export configuration that includes the application client—see Chapter 7, "Exporting Server Modules."

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 export 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. A JCA 1.5 connector can include a message listener. You can receive messages through a connector by mapping its message listener to an MDB.

❖ Importing a connector

Using the Management Console, you can import a connector from a J2EE resource archive (RAR) file.

1. Highlight Connector Modules, right-click, and select Deploy.
2. In the Deploy wizard:
   a. Enter the name of the RAR file, or click Browse and select the file.
   b. Enter a name for the connector module or accept the default, and specify whether to overwrite an existing module with the same name.
   c. Specify whether to validate the deployment descriptors during deployment.
   d. Specify whether to install the module into a server, and select the server.
   e. To save a copy of the archive, enter the location. If the RAR does not include an Ant configuration file, the system generates one automatically, and adds it to the archive copy.
3. The Deploy wizard reads the archive file and creates the connector. Errors are displayed in the status window. Review the status information, then click Close.

When you deploy a connector, the system creates the following connector subfolders:

- Authentication Mechanisms
- Administered Objects
- Connection Factory Types
- Connection Factories

- Message Listener Types

To configure connector properties, see “Configuring connectors” on page 75.

❖ **Exporting a connector**

- To export a connector, create an export configuration that contains the connector—see Chapter 7, “Exporting Server Modules.”

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; the Management Console does not support entity collections. For more information, see Chapter 6, “Using jagtool and jagant,” in the *EAServer Automated Configuration Guide*.

Using EAServer configuration files in J2EE archives

When you deploy J2EE archives, EAServer generates Ant configuration scripts that apply settings from the archive’s deployment descriptor to the new EAServer entities. You can also embed a user configuration file in a J2EE archive to configure settings beyond those that can be specified by the deployment descriptor. For information about creating and configuring Ant scripts and user configuration scripts, see the *EAServer Automated Configuration Guide*.

Undeploying entities

You can undeploy applications, application clients, connectors, EJB packages, and Web applications using the undeploy command line tool. When you undeploy an application, all the entities that it contains are also undeployed. You can also undeploy subentities individually. See undeploy on page 197.

CHAPTER 10 Configuring Java Class Loaders

EAServer uses custom class loaders, which allow you to deploy and update components and applications without restarting the application server.

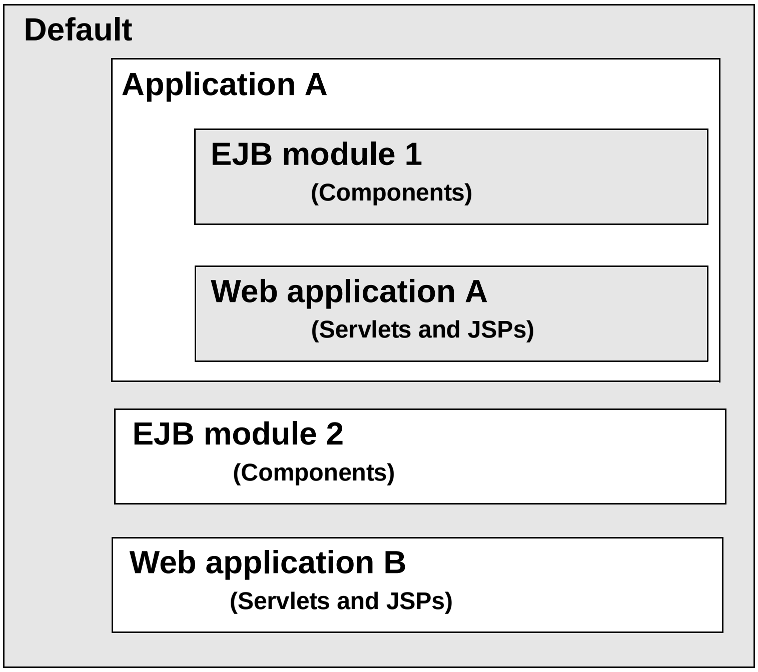
What class loaders do

In Java, a **class loader** loads the Java classes used by an application. Most applications use the Java system class loader, which loads classes from the directories and JAR files specified by the CLASSPATH environment variable. In the normal Java program configuration, you must restart a program or server to begin using updated Java classes. EAServer uses custom class loaders to allow **hot refresh** of Web application classes and Java components without restarting the server.

The class loader hierarchy

Each server, Web application, application, and EJB module has a custom class loader associated with it. EAServer class loaders have a parent-child hierarchy. Every class loader except the predefined default class loader has a parent. This relationship is shown in Figure 10-1.

Figure 10-1: EAServer class loader hierarchy



Class loader delegation policies

The properties of each class loader specify which classes are loaded, the search path to load those classes, and the delegation policy. When a parent and child class loader are configured to load the same class, the delegation policy determines how version conflicts are resolved. The delegation policies are:

- **Parent First** The child class loader delegates to its parent before trying to load the class itself. For example, if you specify Java package X in the class path of an EJB module and the class path of an application that contains the package, the application class loader loads classes in package X.

The Parent-First policy is the default. This setting avoids the overhead incurred by custom-loading multiple copies of the same class among sibling entities. This policy allows sharing of common utility classes used by components and Web applications. It also ensures that both caller and callee are using the same class definition for parameters passed in EJB local interface calls, which is required to avoid ClassCastException errors.

- **Child First** The child class loader tries to load classes itself before delegating to the parent. Use this policy when you must load different versions of a class in different entities. To use this policy, set the Parent-First property to false.

The system class loader

The system class loader is the default class loader provided by the Java runtime. In EAServer, classes that cannot loaded by a custom class loader must be loaded by the system class loader, based on the search order specified by the server's system class path settings. You cannot refresh these classes without restarting the server.

It can be more efficient to configure system class loaders for classes that are used server wide, as long as all components that use them require the same class versions, and you do not need to refresh the classes without restarting the server.

Configuring custom class loaders

When you deploy entities from J2EE archives, EAServer creates a class loader to match the entity name and configures the entities to use the new class loader. You can configure class loaders using the Management Console or an Ant configuration script.

❖ **Configuring class loaders in the Management Console**

1 Connect to the server with the Management Console—see Chapter 2, "Management Console Overview."

2 Expand the Class Loaders folder. Existing class loader definitions display.

- To create a new class loader, right-click the Class Loaders folder and choose Add. Follow the pages in the Add wizard to create the new class loader.
- To modify an existing class loader, click its icon to display the properties in the details pane, modify the settings, then click OK to save the changes.

Table 10-1 describes the class loader properties.

❖ **Using Ant to configure class loaders**

1 Create an Ant configuration file or identify one already in your project where you can add the class loader configuration to the `configure` target. For more information, see Chapter 2, "Ant-Based Configuration," in the *EAServer Automated Configuration Guide*.

2 Add or configure the class loader by invoking the setProperties task from the `configure` or `configure-user target`. For example:

```
<setProperties classLoader="ejbjar-sample">
   <property name="classPath" value="~/ejbjars/*.jar"/>
   <property name=”parentFist” value=”true”/>
   <property name=”parentClassLoader” value=”app-sample”/>
   <property name="classloaderImpl"
value="com.sybase.djc.util.NamedClassLoader"/>
</setProperties>
```

Table 10-1 describes the class loader properties.

Class loader properties

Table 10-1 describes the class loader properties.

Table 10-1: Class loader properties

Management Console	Ant	Description
Class Path	classPath	Specifies additional class path entries for the class loader, in addition to the server class path. Specify entries as a semicolon-separated list. For example: `~/deploy/webapps/foo/WEB-INF/classes;~/deploy/webapps/foo/WEB-INF/lib/**` You can use the following placeholders in entries: • `~`, to specify the EAServer installation directory. • `**`, to specify all JAR files located in a terminal directory.

Management Console	**Ant**	**Description**
Parent Loader	parentClassLoader	Specifies the name of the parent class loader. If not specified, the parent is the "default" class loader.
Parent First	parentFirst	Specifies the default policy for resolving conflicts. If true (selected), the class loader uses the Parent First policy by default. See "Class loader delegation policies" on page 144.
Resolve First Locally	resolveFirstLocally	Lists classes that must be loaded with a child-first policy, overriding the parentFirst setting. Specify a comma-separated list of class names. You can use `**` as a wildcard to represent all classes in a package and other packages that begin with the same prefix. For example, `foo.bar.**` specifies all classes in package foo.bar and classes in packages that begin with foo.bar, like foo.bar.munged, and foo.bar.alt.
Resolve First by Parent	resolveFirstByParent	Lists classes that must be loaded with a parent-first policy, overriding the parentFirst setting. Use the same syntax as for resolveFirstLocally.
Resolve First by System	resolveFirstBySystem	Lists classes that must be loaded by the system class loader. Use the same syntax as for resolveFirstLocally.

Configuring the server's system class path

Classes not loaded by the custom class loader must be loaded by the system class loader, based on the search order specified by the server class path settings described here.

The server class path property

Like other Java programs, the system class loader running in EAServer loads classes in the order specified by the CLASSPATH environment variable. However, you can configure additional entries in the Class Path property, on the General page of the server's property pages in the Web console (or the classPath property if configuring with the Ant setProperties task). For details, see "General tab" on page 24.

The Java Endorsed Standards Override Mechanism

In some cases, you may need to override class versions that are part of the Java runtime core classes. For example, you may need to use a newer version of the XML parsing API (org.xml.*) packages. You cannot override these classes by setting the CLASSPATH environment variable or Class Path server property. Instead, you must use the Java Endorsed Standards Override Mechanism.

The Java Endorsed Standards Override Mechanism allows you to place JAR files in a directory specified by the java.endorsed.dirs Java system property. Classes in the JAR files override like-named classes in the Java core runtime classes. For more information, see Java Endorsed Standards Override Mechanism at http://java.sun.com/j2se/1.4.2/docs/guide/standards/.

To use this mechanism with EAServer, place JAR files containing the classes that override core runtime class versions in the *lib/endorsed* subdirectory of your EAServer installation.

The Java extension mechanism

You can place non-refreshable extension JARS in the *lib/ext* directory. These are added automatically to the class path during server start-up. You can place refreshable extension JARS in the *lib/default/ext* directory. These are loaded by the lib.default-ext class loader, which is a child of the default class loader.

The BOOTCLASSPATH environment variable

The BOOTCLASSPATH environment variable allows you to override class versions that are part of the Java runtime core classes. Sybase recommends that you use the Java Endorsed Standards Override mechanism to override core runtime classes rather than setting this environment variable.

Deciding which classes to add to the custom list

Since system-loaded classes require a server restart to update, use system loading only for classes that never change. For classes that might be refreshed with a component or Web application, add the classes to the custom list. The following sections give more specific guidelines for each entity type.

Custom class lists for EJB components

When you deploy EJB components, classes from the EJB-JAR file are expanded in one of these locations:

- If you are deploying a standalone EJB-JAR file, this directory in the EAServer installation, where *ejbjar* is the base name of the EJB-JAR file:

  ```
  deploy/ejbjars/ejbjar
  ```

- If you are deploying an EJB-JAR file inside an EAR file, this directory in the EAServer installation, where *ear* is the base name of the EAR file and *ejbjar* is the base name of the EJB-JAR file:

  ```
  deploy/ejbjars/ear/ejbjar
  ```

EAServer generates the component’s stub and skeleton classes in the same location.

A Java component’s implementation class and stub classes are automatically part of the custom class list for the component. Add the following additional classes to the custom list:

- Stub classes used for intercomponent calls. For EJB local interface calls, you must also configure sharing of the class instances between the calling entity and the called component.
- Other classes that your component loads and passes as parameters or return values for intercomponent calls, or passes to clients as method return values and output parameter values.
- For EJB components, classes that extend javax.naming.InitialContext or other javax.naming classes and that are called by your component.
- Additional classes that must be reloaded when the component is refreshed. For example, if you are debugging utility classes used by the component, add these classes to the custom list.

Deploy component-specific JAR files in an EAServer subdirectory, such as *java/classes*, and add them to the custom class list.

To configure the custom list, follow the instructions in “Configuring custom class loaders” on page 145.

Custom class lists for Web applications

A Web application’s custom class list must include any classes that must be reloaded when the Web application is refreshed. For example, servlet implementation classes, utility classes called by servlets and JSPs, and stub classes for component invocations should be in the custom class list.

If you deploy classes and JAR files to the standard Web application deployment locations, you need not explicitly list them in the custom class list. The standard Web application deployment locations are under your Web application’s context root, the EAServer subdirectory:

```
deploy/webapps/WebApp
```

Where *WebApp* is the Web application name. Deploy classes and JAR files to these subdirectories of the context root:

- *WEB-INF/classes* for class files

- *WEB-INF/compiled_jsps* for JSP implementation class files
- *WEB-INF/lib* for JAR files that contain archived class files

EAServer automatically adds classes and JAR files that are deployed in the standard locations to the custom class list. Use these locations to deploy classes and JAR files that are specific to your Web application. These directories are equivalent to the like-named directories in a J2EE WAR file.

For classes that are also used in EJB or Java components or other Web applications, you can deploy classes or JAR files in any location, as long as you set each of your class loaders's classPath property to reference the appropriate JARS and directories. For example, if your servlet calls an EJB component, you may want the servlet and component to use the same copies of the component stub classes. You must explicitly list these classes or JAR files in your Web application's custom class list.

Though a component and Web application may custom load the same classes, to enable sharing of one copy of each class, you must configure the same custom class list entries in parent entities up to a common ancestor entity, as described in "Sharing custom-loaded classes" on page 151.

Class loading order for Web applications

EAServer calls classes loaders in the order defined by the class loader hierarchy. For Web applications, class loaders are called in the following order:

1. default
2. lib-default.ext
3. application.*appName*, where *appName* is the application name
4. web.components.*webApp*, where *webApp* is the Web application name
5. system

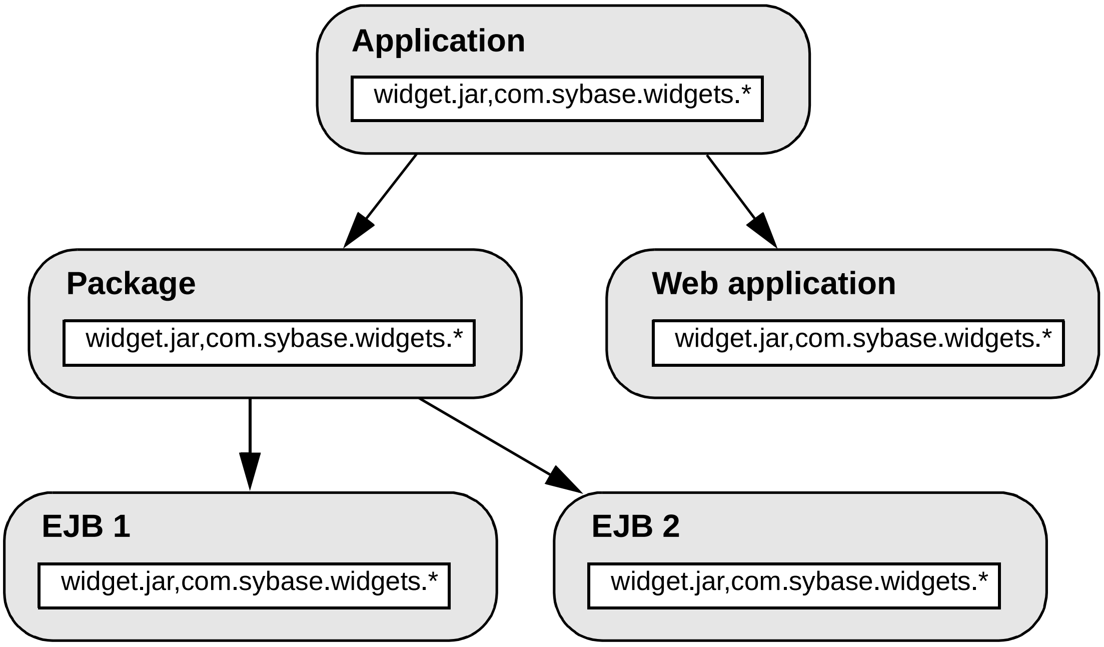
The delegation policy for the web.components.*webApp*, application.*appName*, and lib-default.ext class loaders is Parent First—see "Class loader delegation policies" on page 144.

Sharing custom-loaded classes

You can configure the package-, application-, or server-level class lists to configure sharing of class instances between child entities installed in the package, application, or server. Doing so can decrease server memory use by eliminating copies of custom loaded classes. However, you must also refresh the classes at the level where they are loaded. For example, classes loaded at the server level require a refresh of the entire server.

To configure sharing of custom-loaded classes, first configure the class lists for the Web applications and components that use the class directly, then configure the same entries in the parent entries up to a common ancestor. Make sure the class loader delegation policy is Parent First for entities below this common ancestor. For example, Figure 10-2 shows the entries required to share classes in the JAR file *widget.jar* and the Java package *com.sybase.widgets*, loading one copy of the classes at the application level.

Figure 10-2: Example of sharing custom loaded classes


Application
widget.jar,com.sybase.widgets.*
Package
widget.jar,com.sybase.widgets.*
Web application
widget.jar,com.sybase.widgets.*
EJB 1
widget.jar,com.sybase.widgets.*
EJB 2
widget.jar,com.sybase.widgets.*

Deploy classes that are shared by multiple entities to a common location, and set the classPath property to reference the location.

To configure the server, application, or package class list, follow the instructions in “Configuring custom class loaders” on page 145.

Troubleshooting class loader configuration issues

This section presents additional information that may be useful when you are diagnosing class loader configuration problems.

Commonly encountered problems

Common problems encountered in the custom class list configuration include:

- **Class cast exceptions** In Java, classes loaded by different class loaders are considered different types. You cannot assign a class loaded by one class loader to a reference loaded by another class loader. This restriction must be accounted for when specifying the custom class list, or when deciding the level at which a class should be loaded. Otherwise, the invocation may fail and you see one these Java exceptions in the server log file:

  - java.lang.ClassCastException
  - java.lang.LinkageError
  - java.lang.NoClassDefFoundError
  - java.lang.IncompatibleClassChangeError

  There are two variations of this issue:

  - When using EJB local interfaces, the caller and the callee must share the same instance of classes that are passed as method parameters or return values. In this case, fix the problem by copying the relevant custom class list entries to parent entities, up to a common ancestor. See “Sharing custom-loaded classes” on page 151 for details.

    - For other Java or EJB component calls, the entity that calls the component uses stubs that are system loaded. This call fails because stubs in the component are custom loaded, and Java considers classes that are loaded by different class loaders to be different types, even when the classes have the same name and deployment location. To fix this problem, add the called component's stub classes to the custom class list for the component or Web application that makes the call.

- **Refreshing classes** Classes must be refreshed at the level at which they were loaded. For example, if you configure an application class loader to share some class instances between components and Web applications, you must refresh the application to reload new versions of these classes. You cannot successfully refresh the components, package, or Web application directly.

JAR file locking and copying

JAR files that are in the server's CLASSPATH setting are locked while in use by the system class loader. Consequently, on some platforms such as Windows, you cannot update or overwrite a JAR file while the server is running.

To allow refresh of JAR files that are custom loaded, each class loader instance works with a copy of the JAR files that it has loaded. Each JAR file copy is named *jar-name+sequence.jar*, where *jar-name* is the name of the original JAR file and *sequence* is a sequential number that is used to identify the copy. The JAR file copies are created in the EAServer *temp/server-name/sequence* subdirectory, where *server-name* is the name of your server. EAServer deletes these directories and files when you restart the server.

CHAPTER 11 Runtime Monitoring

This chapter describes how to view server log files, and how to monitor EAServer performance, statistics, and transactions.

Topic	Page
Viewing server log files	155
Monitoring a server	156

Viewing server log files

You can monitor log files for the server to which the Management Console is connected.

❖ Viewing a log file

By default, the *serverName.log* file is created in the *logs* subdirectory. To create other server log files, see “Log/Trace tab” on page 30.

1 Expand the Log Files folder, and select the log you want to view.

  To view archived log files, expand the Old Log Files folder.

2 The text of the selected file displays in the right pane. The File Size field displays the number of lines in the file. Use the following controls to configure the viewing parameters:

  - View All Lines – select the portion of the file to display.
  - Refresh – refresh the display.

Monitoring a server

Monitoring server events and statistics may help you anticipate and prevent server problems. Statistics are available three ways, using the Management Console, the J2EE Management API, or the Jaguar::Monitoring API.

You can download the documentation for the J2EE Management API from the Sun Developer Network at http://java.sun.com/j2ee/tools/management/reference/docs/index.html.

The Jaguar::Monitoring API is available as a CORBA component, but is provided only for backward compatibility. EAServer 6.0 provides limited support for the last maximum and peak maximum statistics. If 60-second statistics are enabled, the value of last maximum is the same as the 60-second maximum value. In most cases, the value of peak maximum is the same as the current value. You can enable 60-second statistics in the Management Console, on the server's General tab—see "General tab" on page 24. The HTML documentation for the Jaguar::Monitoring API is in the *html/ir* subdirectory of your EAServer installation.

Memory monitoring enables EAServer to prevent low memory from causing the server to crash. Two SocketListener properties support this feature, memoryAlarmLevel and memoryCriticalLevel. memoryAlarmLevel defines the percentage of system memory that can be used before EAServer notifies users of low memory. Memory usage equal to the value of memoryCriticalLevel prevents new component invocations.

Performance monitoring looks at response times for components in the server. If you enable performance monitoring for a component, the server calculates the average time it takes to execute methods on this component. If the average time is too long (as configured by the user), EAServer blocks subsequent requests for this component. You can monitor performance at the component, listener, and server level. Listener-level monitoring takes precedence over component-level monitoring, and server-level monitoring takes precedence over both.

Component-level monitoring

To configure performance monitoring at the component level, add the performanceMonitor attribute to the configuration file for the component you want to monitor. The performanceMonitor attribute takes two parameters:

- maxResponseTime – the maximum response time (in milliseconds) before blocking occurs.
- minInstances – the minimum number of active instances.

For example, if the maximumResponseTime is 2000 ms and minInstances is 10, and if the response time for this component is more than 2000 ms, EAServer allows only 10 instances of the component to run concurrently. The following is a sample EJB-JAR configuration file that sets these values for the performanceMonitor:

```
<setProperties
   component="ejb.components.crescalator.crhandler.EscalationLocalHome"
   merge="false">

   <property name="ejbName" value="Escalation"/>
   <instancePool timeout="${ejb.poolTimeout}"/>
   <permitAccess ports="${ejb.allowedPorts}"/>
   <classLoader name="ejb.components.crescalator.crhandler"/>
   <threadMonitor name="${ejb.localThreadMonitor}"/>
   <transaction type="Required" retry="${ejb.transactionRetry}"
      sqlIsolationLevel="${sql.isolationLevel}"/>
   <performanceMonitor maxResponseTime="2000" minInstances="10"/>

</setProperties>
```

Listener and server monitoring

To configure performance monitoring at the listener or server level, set the performanceMonitor attribute in the entity’s Ant configuration file, following the example above.

A value of -1 disables monitoring at the level it is defined. The default monitoring configuration is:

- At the server and listener levels, maxResponseTime is set to -1; therefore, performance monitoring is not performed.
- At the server level, memory monitoring is set to a positive value, and at the listener level, the setting is -1; therefore, memory monitoring is performed at the server level.

Viewing server statistics

To view server statistics, you can use any of the following methods:

- Use the Management Console to view statistics for specific entity types.
- From the Management Console, select to display statistics in a spreadsheet format.

- Use a spreadsheet program, such as Microsoft Excel, and import statistics. For example, from Excel, select Data | Import External Data | New Web Query. As the URL, enter:

  ```
  http://host:port/wsh/run?command=get-statistics
  ```

  where *host* and *port* are the server's host name and HTTP port, respectively.
- View statistics in text files in the server's *logs/statistics* subdirectory.

Some statistics are not collected by default, such as memory usage statistics. To see these statistics, enable the scheduled tasks that record them as described in "Scheduled tasks for statistics collection" in Chapter 1, "Introduction," in the *EAServer Performance and Tuning Guide*.

The remainder of this chapter describes using the Management Console to view statistics.

❖ Viewing statistics in spreadsheet format

1. Expand the icon for the server you want to monitor.
2. On the General tab, select the Statistics URL.
3. In the Login dialog, enter the administrative user name and password for the Management Console.

   A new window displays statistics for all the server's entities.

❖ Viewing statistics on entity tabs

1. Expand the icon for the server you want to monitor.
2. Highlight the Statistics folder. Statistics display in the right pane on the:
   - CheckMemoryUsageTask tab – memory use statistics.
   - HTTPStatsManager tab – HTTP connection statistics.
   - IIOPStatsManager tab – IIOP connection statistics.
   - ScheduledTask tab – scheduled task statistics.
   - ThreadMonitor tab – thread pool activity.
   - DataSource tab – data source connections.
   - InstancePool tab – component instance pool activity.
   - MessageQueue tab – statistics for message queue activity.
   - MessageTopic tab – statistics for message topic activity.

- MethodProfiler tab – method invocations.
- TransactionContext tab – transaction contexts.
- TransactionManager tab – transactions.

Each tab displays *counters*, which represent monitored statistics, such as the number of component invocations or HTTP requests. When you select a tab, the current values of the counters display. 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.
- *High-value counters* represent the maximum value since starting the server.
- *Low-value counters* represent the minimum value since starting the server.

3 To refresh or change the counters view, choose from:

- **Current Values** Display the current values; this is the default.
- **60 Second Summary** Display a summary of the values that pertain to the last 60 seconds.
- **60 Minute Summary** Display a summary of the values that pertain to the last 60 minutes.
- **Refresh** Refresh the display to obtain the latest counters.

Some statistics are valid for a particular moment in time, so you must refresh the display to get the most current information.

CheckMemoryUsageTask tab

EAServer collects the following memory statistics:

Statistic	Description
Java Free Memory	The number of bytes of free memory
Java Maximum Memory	The maximum number of bytes of virtual memory that the Java virtual machine (JVM) can use
Java Total Memory	The number of bytes of virtual memory that are currently allocated for use by the JVM
Process Maximum Memory	The maximum amount of virtual memory that the server can use, specified in bytes
Process Total Memory	The maximum number of bytes of virtual memory that can be used by the server, including the JVM

HTTPStatsManager tab

EAServer collects the following HTTP connection statistics:

Statistic	Description
Accepted Connections	The number of HTTP connection requests that were accepted by the server
Number of Active Requests	The number of requests that are currently active
Average Requests per Connection	The average number of requests for each HTTP connection
Number of Bytes Written	The total number of bytes written for all HTTP connections since starting the server
Average Duration of Connections	The average duration (in milliseconds) of HTTP connections
Number of Errors	The number of HTTP connection requests that resulted in an error
Open Connections	The number of HTTP connections that are open
Average Duration of Requests	The average duration (in milliseconds) of HTTP requests
Number of Requests	The total number of HTTP requests

IIOPStatsManager tab

EAServer collects the following IIOP connection statistics for a server:

Statistic	Description
Accepted Connections	The number of IIOP connection requests that have been accepted by the server
Open Connections	The number of IIOP connections that are open

ScheduledTask tab

EAServer collects the following scheduled task statistics:

Statistic	Description
Execute Time	The average number of milliseconds the task runs
Success Count	The number of times the task has been run since starting the server

ThreadMonitor tab

Because thread monitor statistics are captured for active threads at a particular moment, you must refresh the display to see changes in the activeThreadCount.

Statistic	Description
Active Thread Count	Total number of active threads
Current Maximum Active Threads	Maximum threads allowed for the thread monitor

DataSource tab

The DataSource tab displays statistics for cached database connections.

Statistic	Description
Create Count	The number of data source connections that have been created
Close Count	The number of data source connections that have been closed
Pool Size	The maximum number of connections allocated to the pool for this data source
Free Pool Size	The number of connections that are not in use
Waiting Thread Count	The number of requests waiting for a connection
Error Count	The number of connection requests that have resulted in an error

Statistic	Description
Connect Time	The total amount of connection time for all connections, specified in milliseconds
Use Time	The total amount of time connections have been active, specified in milliseconds
Wait Time	The total amount of connection wait time, specified in milliseconds
Open Count	The number of open connections

InstancePool tab

Instance pool activity includes the following statistics:

Statistic	Description
Active Count	The number of active instances in the pool
Discard Count	The number of instances that have been discarded from the pool
Passive Count	The number of instances in the pool that are idle.
Pool Size	The number of instances in the pool, active or inactive
Pool Timeout	The number of instances that have timed out of the pool

MessageQueue tab

EAServer gathers the following message queue statistics:

Statistic	Description
Acknowledge Count	The number of messages that have been acknowledged
Add Count	The number of messages that have been added to the queue
Discard Count	The number of messages that have been discarded from the queue; transient messages are discarded if a queue either overflows or times out
Timeout Count	The number of message that have been discarded because the queue timed out
Expired Count	The number of messages that have expired
Receive Count	The number of messages the queue has received
Recover Count	The number of messages the queue has recovered
Send Count	The number of messages that have been sent to the message queue

Statistic	Description
Messages in Memory Count	The number of messages that are currently in memory
Waiting for Acknowledge Count	The number of messages that are waiting for acknowledgement
Delivery Time	The total number of milliseconds spent delivering messages to the queue

MessageTopic tab

EAServer gathers the following message topic statistics:

Statistic	Description
Acknowledge Count	The number of messages that have been acknowledged
Add Count	The number of messages that have been added to the topic's subscription queue
Discard Count	The number of messages that have been discarded from the topic's subscription queue
Receive Count	The number of messages that have been received by the topic's subscription queue
Recover Count	The number of messages that have been recovered by the topic's subscription queue
Publish Count	The number of messages published to this topic
Timeout Count	The number of messages that have been published to the topic
Messages in Memory Count	The number of messages in the topic's subscription queue's memory
Waiting for Acknowledge Count	The number of messages waiting for acknowledgement by the topic's subscription queue
Durable Subscription Count	The number of durable subscriptions for the topic
Delivery Time	The total number of milliseconds spent delivering messages to the topic

MethodProfiler tab

This tab displays statistics for method invocations; the number of times each method has run, the minimum and maximum times it took to invoke the method, and the total amount of time the method has run since the server started. You can view statistics for all components installed in the server, including those that are installed indirectly as part of an installed application.

Statistic	Description
Method Time	The number of milliseconds for each time category
Exception Count	The number of exceptions the method has thrown

You can also view method statistics by accessing the Statistics node under specific entities—see “Monitoring Web application and EJB statistics” on page 165.

TransactionContext tab

The TransactionContext tab displays statistics about the operations that comprise incomplete transactions for a server.

Statistic	Description
Commit Count	Total number of committed transactions since the server started
Transaction Time	The total number of milliseconds that all transactions have taken to complete

TransactionManager tab

The TransactionManager tab displays statistics about incomplete transactions for a single server. Because statistics are captured for incomplete transactions at a particular moment, you must refresh the display to see changes in the incomplete transaction status.

Statistic	Description
Commit Count	The number of committed transactions since the server started

Monitoring CPU usage graphically

To view a graphical display of the server's CPU usage:

1. Expand the server to monitor.
2. Select Statistics – CPU. The graph displays in the right pane.
3. To change the number of seconds between refreshing the view, set the refresh rate, then select Apply.

Monitoring server components graphically

To monitor the server components graphically:

1. Expand the server to monitor.
2. Select Statistics – Components. The graph displays in the right pane.
3. Use the following controls to configure the viewing parameters:
    - Width – the pixel width of the monitor.
    - Height – the pixel height of the monitor.
    - Refresh Rate – the number of seconds between refreshing the view.

    If you make changes, select Apply.

Monitoring Web application and EJB statistics

You can view statistics for Web applications and EJBs by accessing the Statistics node beneath an individual entity.

❖ **Monitoring Web application statistics**

1. Expand these folders: Web Applications | *WebApp* | Web Component | *compName*, where *WebApp* is the name of the Web application and *compName* is the name of the Web application component to monitor.
2. Select the Statistics node. In the right pane, statistics display for this component.

❖ **Monitoring EJB component statistics**

1. Expand these folders: EJB Modules | *package* | EJB Components | *compName*, where *package* is the package name and *compName* is the name of the EJB component to monitor.

2 Select the Statistics node. In the right pane, statistics display for this component.

CHAPTER 12 Command Line Tools

This chapter documents the EAServer command line tools.

Overview

Unless otherwise specified, all commands are located in the *bin* directory of your EAServer installation.

In addition to the detailed documentation provided here, you can see a syntax summary for any command by specifying `-help` as the sole option, for example:

```
deploy -help
```

Note Command line tools end with the *.bat* suffix on Window platforms (which you can omit). UNIX and Linux commands end with the .sh suffix (which you must include in your command).

Entity references

Several commands, such as export and undeploy, require that you specify the entity type and name. You must specify entity references in a format matching the entity type as listed below:

- `appclient-`***`client`*** – a J2EE application client named *client*.
- `application-`***`appname`*** – a J2EE application named *appname*.
- `connector-`***`conn`*** – a connector or resource adaptor named *conn*.
- `ejbjar-`***`pack`*** – an EJB module named *pack*.

- `webapp-`***`wname`*** – a Web application named *wname*.

If a command allows you to specify multiple entity names, you can use wildcards. For example, use `ejbjar-foo*` to specify all EJB modules that begin with `foo`, or `appclient-*` to specify all application clients.

Command line tool security

Security roles provide a fine-grained ability to control who can execute which commands. These roles are supported for wsh (see wsh on page 200):

- wsh.run.server.*xxx*
- wsh.run.system.*xxx*

  where .*xxx* is the optional name of the command.

wsh commands are divided into two types, server and system. The server commands consist of:

- login
- logout
- get-statistics
- ping-server
- server-status
- refresh
- refresh-server
- restart-server
- stop-server
- start-module
- stop-module

The following is a partial list of the system commands. However, system commands also include commands that are executed as child processes, which include all of the commands in the *bin* subdirectory of your EAServer installation, such as deploy.bat (Windows) or deploy.sh (UNIX):

- cat
- ls

- mkdir
- rm
- rmdir
- extract
- undeploy

If you have the wsh.run.server role, you can execute any server command. If you have the wsh.run.server.get-statistics role, and no other wsh.run roles, then you can run only the get-statistics command. This also applies to the wsh.run.server roles.

These roles are supported for wfs (see wfs on page 198):

- wfs.put.server.*xxx*
- wfs.put.system
- wfs.get.server.*xxx*
- wfs.get.system
- wfs.delete.server.*xxx*
- wfs.delete.system
- wfs.view.server.*xxx*
- wfs.view.system

where *.xxx* is the name of the subdirectory relative to the EAServer installation directory.

Keep in mind that:

- A server file is any file inside the context of the EAServer installation directory.
- A system file is any file that is not under the EAServer installation directory.

If a user has the wfs.put.server.html role, he or she can put files into the EAServer *html* subdirectory. If users have the wfs.put.server role, they can put files into any EAServer subdirectory. If users have the wfs.put.system role, they can put files into any directory outside EAServer. System roles do not support the optional directory because it is unclear as to which file system that directory refers.

By default the admin user has these roles:

- wsh.run.server
- wsh.run.system
- wfs.put.server
- wfs.put.system
- wfs.get.server
- wfs.get.system
- wfs.delete.server
- wfs.delete.system
- wfs.view.server
- wfs.view.system

Commands

The section lists the commands in alphabetical order.

Each command has its own heading and each command section contains a description of the command, its syntax, a list of options, and an example of its use.

asa-init

Description	Allows EAServer data sources to initialize Adaptive Server Anywhere, the location of which is defined by asa-setenv.
Syntax	asa-init

asa-setenv

Description

Sets the default Adaptive Server Anywhere environment variable that allows EAServer data sources to set up the location of ASA9. Edit the file to change the location.

If ASA9 is already installed, the global environment should already be set. In this case, comment out the contents of asa-setenv.

Syntax

```
asa-setenv
```

asa-start

Description

Allows EAServer data sources to start Adaptive Server Anywhere, the location of which is defined by asa-setenv.

Syntax

```
asa-start
```

asa-stop

Description

Allows EAServer data sources to stop Adaptive Server Anywhere, the location of which is defined by asa-setenv.

Syntax

```
asa-stop
```

clear-password-history

Description

Clears the password history for the given user.

Syntax

```
clear-password-history <username>
```

Option	Description	Required
username	The user for which you are clearing password history, in the form of username@domain.	Yes

Examples

This example clears the password history for the user “guest” that belongs to the test domain:

```
clear-password-history guest@test
```

clear-protected-property

Description

Deletes the value for the given property from protected storage (usually a password).

Syntax

clear-protected-property <*property*>@<*component*>:<*instance*>

Option	Description	Required
property		Yes
component		Yes
instance		Yes

Examples

This example clears property protection for this password:

```
clear-protected-property
password@com.sybase.djc.sql.DataSource:default
```

See also

set-protected-property

config-jacc

Description

Configures the JACC target for the corresponding Web application.

Syntax

config-jacc *target*

Option	Description
target	The name of the Web application or EJB-JAR file for which JACC is configured

Examples

This example configures JACC for webapp-myweb:

```
config-jacc webapp-myweb
```

See also

delete-jacc
"JACC (JSR-115) support" on page 84 in the *EAServer Security Administration and Programming Guide*

configure

Description	Allows you to run Ant scripts in the *config* directory. For example, you could run the configure target in an entity's configuration file to apply changes to the entity properties.
Syntax	configure *entity* [*ant-opts*]

Option	Description
entity	The name of the entity to configure, in the format described under "Entity references" on page 167. entity can also be the prefix (without *.xml* suffix) of any Ant configuration script in the *config* directory. It does not need to be a script for a deployed module.
ant-opts	Any additional command line options to be passed to Ant.

See also

recompile, refresh

Chapter 2, "Ant-Based Configuration," in the *Automated Configuration Guide*.

delete-jacc

Description	Deletes the JACC target for the corresponding Web application.
Syntax	delete-jacc *target*

Option	Description
target	The name of the Web application or EJ-JAR file for which JACC is deleted

Examples

This example deletes JACC for webapp-myweb:

```
delete-jacc webapp-myweb
```

See also

config-jacc
"JACC (JSR-115) support" on page 84 in the *EAServer Security Administration and Programming Guide*

deploy

Description — Deploys a J2EE archive to the server.

Syntax

deploy
[-jdk15]
[-package *package-name*]
[-eas5naming *true-false*]
[-jspc *true-false*]
[-disableValidation *true-false*]
[-overwrite *true-false*]
[-server *host*:*port*]
[-copyWithConfig *destDir*]
[-noidl]
modules
[-jacc]

Option	Description
`-jdk15`	If specified, EAServer compiles generated Java source files with the JDK 1.5 compiler. Otherwise, JDK 1.4 is used.
`-server` *`host`*`:`*`port`*	Remotely deploys the archive to the server running on the specified host and listening on the specified HTTP port. If you do not specify a server, the archive deploys locally to the same installation from where you run the deploy command. You must have an active wlogin session with the server, using an account that has Administrator role privileges on that server. **Note** Instead of using the *host:port* syntax, you can pass the server name, if it is defined in the local repository.
`-package` *`package-name`*	When deploying a single archive, this option overrides the default package name, which is the base name of the archive file (unless you specify `-eas5naming true`, in which case the default is the base name of the archive's deployment descriptor file).
`-package:`*`pkg-name new-name`*	When deploying an EAR or multiple archives of any format, this option overrides the default package name for the specified archive. For example, if an EAR file contains an EJB-JAR file *foo.jar*, `-package:foo bar` specifies that the EJBs in *foo.jar* must be deployed in the package `bar` rather than using the default package name of `foo`.
`-eas5naming` *`true-false`*	Causes deploy to select default package names from the base name of the deployment descriptor file within the archive. The default is false, which means package names default to the base name of the archive.
`-jspc` *`true-false`*	When deploying a Web application, whether to compile the JSPs it contains. The default is true.

Option	Description
`-overwrite` *`true-false`*	Whether to overwrite existing entities with the same name as those in the archive. In the as-installed configuration, the default is true. You can change the default as described in "Configuring deployment defaults" on page 177.
`-copyWithConfig` *`destDir`*	If specified, a copy of the archive file is created in the EAServer installation in the specified destination directory. The copy is identical to the source, except that an EAServer configuration file is added to the *META-INF* directory. The destination directory must exist and be a full path or a path relative to the current directory.
`-contextPath` *`path`*	When deploying a WAR file, the context path for the deployed Web application. If not specified, the default is the base name of the WAR file converted to all lowercase letters.
`-jsr154filter` *`true-false`*	When deploying a Web application, whether to enable the JSR 154 support filter in the Web application. The JSR 154 filter provides a more strict interpretation of the Servlet 2.4 specification. Specifically: • Support for request attribute listeners. These are not allowed unless the JSR 154 filter is enabled. • Support for dispatching of chained requests with "wrap under" rather than "wrap over" semantics, as described in the Jetty documentation at http://jetty.mortbay.org/jetty/doc/servlet24.html#d0e711. If the JSR 154 filter is disabled, chained requests use "wrap over" semantics. The default is false.
`-parentCl` *`classloader-name`*	Overrides the default parent class loader. The default depends on what you are deploying: • For EJB components and Web applications deployed in an EAR file, the default parent class loader is the class loader of the parent application. • For Web applications deployed in a standalone WAR file, the default parent is lib.default-ext. • For EJB components deployed in a standalone EJB-JAR file, the default parent is the default class loader.
`-disableValidation` *`true-false`*	Specifies whether to validate J2EE deployment descriptors. In the as-installed configuration, the default of false enables validation as required by the J2EE specification. You can change the default as described in "Configuring deployment defaults" on page 177.
`-keepModuleOnFailure` *`true-false`*	Specifies whether to remove or keep partially deployed files when deployment fails. In the as-installed configuration, the default is false, which causes partially deployed files to be removed if the deployment fails. You can change the default as described in "Configuring deployment defaults" on page 177.

Option	Description

`-noidl`	If specified, disables generation of CORBA IDL when deploying EJB-JAR files or EAR files that contain EJB-JAR files. CORBA IDL is required to invoke EJB components from CORBA or PowerBuilder clients. Deployment may be slightly faster if IDL generation is disabled.
	
	You disable IDL generation for all deployments by running the command:
	
	`configure idl-generator-off`
	
	To enable IDL generation, run the following command. You can still disable IDL generation by specifying the `-noidl` command-line option:
	
	`configure idl-generator-on`

modules	The path and name to one or more J2EE archive files. You can specify multiple archives, separated by commas. The following extensions and formats are allowed: • *ear* – other archive formats bundled inside an enterprise EAR file. Deploys as an application. • *jar* – an EJB-JAR or J2EE application client archive. • *rar* – a J2EE connector or resource adaptor archive. • *war* – a Web application archive.
`-jacc`	If specified, EAServer configures the deployed module to use a custom JACC (Java Authorization Contract for Containers) implementation to enforce authorization constraints, rather than the default role-based access control mechanism.

In addition to the above options, the following table describes deployment options used to deploy an EJB as a Web service. See Chapter 4, “Web Services Administration” for more information.

Option	Description
-*contextPath path*	Specifies the context path for Web application deployment. The default is the name of the WAR file. If the -package option is specified, then the package is the context path.
-*ws*	Exposes any stateless session beans with a remote interface as a Web service.
-*ws:ejbName*	Exposes the stateless session bean *ejbName* as a Web service.
-*ws:jarFileInEar:ejbName*	Exposes the stateless session bean *ejbName* in jarFile in an application as a Web service.
-*wscontextpath*	Specifies the context path for an EJB Web service.
-*wscontextpath:jarFileInEar:contextpath*	Specifies the context path for an EJB Web service in an application EAR.

Option	Description
-*wsstyle* <style>	Specifies the style (DOCUMENT, RPC or WRAPPED) of the binding in the generated WSDL file when you expose an EJB as a Web service.
-*wsstyle:jarFileInEar:style*	Specifies the style (DOCUMENT, RPC or WRAPPED) of the binding in the generated WSDL file when you expose an EJB as a Web service in an application EAR.
-*wsuse* <use>	Specifies the use (LITERAL or ENCODED) of items in the binding for the generated WSDL file when you expose an EJB as a Web service.
-*wsuse:jarFileInEar:use*	Specifies the use (LITERAL or ENCODED) of items in the binding for the generated WSDL when you expose an EJB as a Web service in an application EAR file.
-*wswebappname* <webappname>	Overrides the default Web application name generated from an EJB Web service.
-*wswebappname:jarFileInEar:webappname*	Overrides the default Web application name generated from an EJB Web Service in an application EAR file.

Configuring deployment defaults The configuration file *deploytool-options.xml* in the EAServer *config* subdirectory configures defaults for some deployment settings, such as the default values for the `-overwrite`, `-disableValidation`, and `-keepModuleOnFailure` options. To change the defaults, edit the configuration file and apply the settings with the Management Console or the following configure command:

```
configure deploytool-options
```

See also

export-config, undeploy

djc-ant

Description

Runs the Apache Ant installation that is included with EAServer.

Syntax

Same as Ant. For details, see the documentation for Ant 1.6.5, located on the Apache Web site at http://ant.apache.org/.

Usage

This script is simply a wrapper that runs Ant in the environment configured by djc-setenv.

djc-setenv

Description

Configures environment settings required to run the server and build components, such as PATH, CLASSPATH, and so forth.

Syntax

Do not run this script directly from the command line. You can call it from your own build or run scripts to configure the environment.

Customizing the environment To customize the environment, do not modify the installed copy of the djc-setenv script. Instead, create the script local-setenv.bat (for Windows platforms) or local-setenv.sh (for UNIX platforms).

In your local-setenv script, configure any alternate or additional settings required. Note the following:

- Do not overwrite environment variables that configure binary, library, or Java class search paths such as CLASSPATH, PATH, LD_LIBRARY_PATH, SHLIB_PATH, and so forth. Instead, prepend or append any entries that you require to the existing setting.
- Check the value of the DJC_SETENV_WAS_SET environment variable to ensure that your settings are not applied more than once. Failure to do so can cause the PATH, CLASSPATH, or other variables to grow beyond the operating system length limit.

  On Windows platforms, in local-setenv.bat, use the syntax shown in this example:

```
if "%DJC_SETENV_WAS_SET%" == "true" goto END
  set CLASSPATH=%CLASSPATH%;i:/j2eetck1.4/lib/cts.jar
  set CLASSPATH=%CLASSPATH%;i:/j2eetck1.4/lib/tsharness.jar
  set CLASSPATH=%CLASSPATH%;i:/j2eetck1.4/lib/tspackager.jar
  set CLASSPATH=%CLASSPATH%;i:/j2eetck1.4/lib/tsprovider.jar
  set CLASSPATH=%CLASSPATH%;i:/j2eetck1.4/classes
:END
```

On UNIX platforms, in local-setenv.sh, use the syntax shown in this example:

```
if [ ! "$DJC_SETENV_WAS_SET" = "true" ]; then
  set CLASSPATH=$CLASSPATH:/my/other/stuff.jar
  export CLASSPATH
fi
```

ejb-login

Description

Provides single-sign-on access to a named EJB provider.

Syntax

ejb-login <*provider-instance-name*>

Option	Description	Required
provider-instance-name	The instance name of the component com.sybase.ejb.client.EjbProvider which contains the client side EJB configuration. The default value is "default".	Yes

Prerequisites Before you can use the ejb-login or ejb-logout scripts, you must define an instance of the com.sybase.ejb.client.EjbProvider component for each EjbProvider you want to use. The following default configuration script located in *${djc.home}/config/default-ejb-providers.xml* shows you how to do this. For example:

```
configure default-ejb-providers
```

The output looks similar to:

```
Buildfile: M:\target1.4\bin\..\config\default-ejb-
providers.xml

configure:
BUILD SUCCESSFUL
Total time: 1 second
```

and creates the following repository component instance definition:

```
cat Repository\Instance\com\sybase\ejb\client\EjbProvider\default.properties
```

```
#Instance Properties
java.naming.factory.initial=com.sybase.ejb.client.InitialContextFactory
java.naming.provider.url=iiop\://USER-PC\:2000
java.naming.security.principal=user
ant.project=default-ejb-providers
java.naming.security.credentials=*
```

Examples

This example logs you in to a session of the default EJB:

```
ejb-login default
```

Which prompts you for a user name and password:

```
Username (testuser): admin@system
Password:XXXXX
```

And indicates if the login was successful:

```
Logged in.
```

See also ejb-logout

ejb-logout

Description Logs out from a single-sign-on session of a named EJB provider.

Syntax ejb-logout <*provider-instance-name*>

Option	Description	Required
provider-instance-name	The instance name of the component com.sybase.ejb.client.EjbProvider which contains the client side EJB configuration. The default value is "default".	Yes

Prerequisites Before you can use the ejb-login or ejb-logout scripts, you must define an instance of the com.sybase.ejb.client.EjbProvider component for each EjbProvider you want to use. The following default configuration script located in *${djc.home}/config/default-ejb-providers.xml* shows you how to do this. For example:

```
configure default-ejb-providers
```

The output looks similar to:

```
Buildfile: M:\target1.4\bin\..\config\default-ejb-
providers.xml

configure:
BUILD SUCCESSFUL
Total time: 1 second
```

and creates the following repository component instance definition:

```
cat Repository\Instance\com\sybase\ejb\client\EjbProvider\default.properties
```

```
#Instance Properties
java.naming.factory.initial=com.sybase.ejb.client.InitialContextFactory
java.naming.provider.url=iiop\://USER-PC\:2000
java.naming.security.principal=user
ant.project=default-ejb-providers
java.naming.security.credentials=*
```

Examples This example logs you out of a session of the default EJB:

```
    ejb-logout default
```

And indicates if the logout was successful:

```
  Logged out.
```

See also ejb-login

export

Description Packages export configuration ZIP files for the specified entities.

Syntax export [-exportDir *path*] [-exportGenFiles] [-server *host*:*port*] *entities*

Option	Description
`-exportDir` *path*	Creates archives in the specified directory. If not specified, the default is the *html/download* directory of the EAServer installation. If you are exporting from a remote server, the path must be valid in the file system of that server. If not specified, the default is *html/download* in the EAServer installation, which allows you to download the export ZIP file using a Web browser or other HTTP client.
`-exportGenFiles`	If you specify this option, the export configuration includes generated files such as stubs and skeletons for EJBs and compiled servlet classes for JSPs.
`-server` *host*:*port*	Remotely exports the archive from the server running on the specified host and listening on the specified HTTP port. If you do not specify a server, the archive exports locally from the same installation where you run the command. You must have an active wlogin session with the server, using an account that has Administrator role privileges on that server. **Note** Instead of using the *host:port* syntax, you can pass the server name, if it is defined in the local repository.
entities	The names of the entities to be packaged as ZIP files, in the format described under "Entity references" on page 167. export creates a ZIP file containing an export configuration for each entity.

See also Chapter 7, "Exporting Server Modules"

export-config

Description Creates or updates an export configuration.

Syntax export-config -action *action* [-verbose] *name*

Option	Description
`-action` *`action`*	The action to perform: • `build` to create a ZIP file containing the files required to synchronize the export configuration to other servers. • `synchronize` to replicate the export configuration by deploying it to the servers specified in the export configuration properties.
`-verbose`	Run in verbose mode.
name	The name of the export configuration to build or synchronize.

See also Chapter 7, "Exporting Server Modules"

idl-compiler

Description Generates code for binding CORBA interface clients to IDL interfaces and datatypes.

Syntax

```
idl-compiler prefix-opts idl-files format-opts suffix-opts
```

where:

- **prefix-opts** A list of one or more of these options:

Option	Description
`-v`	Run in verbose mode.
`-i folder`	Add folder for IDL include files. The default is the EAServer *Repository/IDL* subdirectory.
`-r folder`	*Required.* Specify the location of the EAServer Repository subdirectory, for example, *d:\EAServer\Repository*.

Option	Description
`-lang language`	Specify a language for error messages. Possible values are: • `us_english` (the default) • `english` • `japanese` • `german` • `french` • `korean` • `chinese`

- **idl-files** A list of one or more IDL module files, separated by spaces. Nested IDL modules are organized in nested directories; specify the path to a nested module relative to the EAServer *Repository/IDL* subdirectory. For example, the file name that defines the module *com::foo::bar* is *com/foo/bar.idl*.
- **format-opts** A list of one or more of the following options:

Option	Description
`-cr`	Use CR/LF (carriage return/line feed) in generated files.
`-f folder`	Specify base folder for code generation.
`-jp IM=JP`	Set Java package *JP* for IDL module *IM*, for example: `-jp com::foo::bar=com.foo.bar` The specified package overrides the default. See “Specifying Java package mappings for IDL modules” in Chapter 3, “Using CORBA IDL,” in the *CORBA Components Guide*.
`-nh`	Suppress generation of Java helper and holder classes.
`-np`	Suppress module prefix for PowerBuilder stubs.

- **target-opts** A list of one or more of the following options:

Option	Description
`-cpp`	Generate C++ stubs following CORBA 2.3 C++.language bindings specification.

Option	Description
`-java`	Generate Java interfaces and parameter datatype classes according to the CORBA 2.3 Java.language bindings specification. **Note** idl-compiler does not generate stub implementation classes. The Java CORBA client interface runs on top of the EJB client interface. After generating the interface classes with idl-compiler, run stub-compiler to generate EJB stubs.
`-pb`	Generate code for CORBA 2.3 / PowerBuilder.
`-idl`	Generate pretty-printed IDL output.
`-html`	Generate HTML documentation for the types and interfaces defined in the specified IDL modules.

See also

stub-compiler

Chapter 3, "Using CORBA IDL," in the *CORBA Components Guide*

jaguar-compiler

Description

Compiles a CORBA package contained on EAServer.

Syntax

jaguar-compiler <-server:servername> <packageName>

Option	Description	Required
server	The server calling the compiler. jaguar-compile uses the server name to find the server properties used to determine deployment options.	Yes
packageName	The CORBA component package name.	Yes

Examples

In this example, EAServer1 calls the compiler for testpackage:

```
jaguar-compiler -server:EAServer1 testpackage
```

jms-login

Description

Provides single-sign-on access to a named JMS provider.

Syntax

jms-login <*provider-instance-name*>

Option	Description	Required
provider-instance-name	The instance name of the component com.sybase.jms.client.JmsProvider which contains the client side JMS configuration. The default value is "default".	Yes

Prerequisites Before you can use the jms-login or jms-logout scripts, you must define an instance of the com.sybase.jms.client.JmsProvider component for each JmsProvider you want to use. The following default configuration script located in *${djc.home}/config/default-jms-providers.xml* shows you how to do this. For example:

```
configure default-jms-providers
```

The output looks similar to:

```
Buildfile: M:\target1.4\bin\..\config\default-jms-providers.xml
configure:
BUILD SUCCESSFUL
Total time: 1 second
```

and creates the following repository component instance definition:

```
cat Repository\Instance\com\sybase\jms\client\JmsProvider\default.properties
```

```
 #Instance Properties
java.naming.factory.initial=com.sybase.jms.client.Init
ialContextFactory
java.naming.provider.url=iiop\://USER-PC\:2000
java.naming.security.principal=user
ant.project=default-jms-providers
java.naming.security.credentials=*
```

Examples

This example logs you in to a session of the default JMS:

```
jms-login default
```

Which prompts you for a user name and password:

```
Username (testuser): admin@system
Password:XXXXX
```

And indicates if the login was successful:

```
Logged in.
```

See also jms-logout

jms-logout

Description Logs out from a single-sign-on session of a named JMS provider.

Syntax jms-logout <*provider-instance-name*>

Option	Description	Required
provider-instance-name	The instance name of the component com.sybase.jms.client.JmsProvider which contains the client side JMS configuration. The default value is “default”.	Yes

Prerequisites Before you can use the jms-login or jms-logout scripts, you must define an instance of the com.sybase.jms.client.JmsProvider component for each JmsProvider you want to use. The following default configuration script located in *${djc.home}/config/default-jms-providers.xml* shows you how to do this. For example:

```
configure default-jms-providers
```

The output looks similar to:

```
Buildfile: M:\target1.4\bin\..\config\default-jms-providers.xml
configure:
BUILD SUCCESSFUL
Total time: 1 second
```

and creates the following repository component instance definition:

```
cat Repository\Instance\com\sybase\jms\client\JmsProvider\default.properties
```

```
 #Instance Properties
java.naming.factory.initial=com.sybase.jms.client.Init
ialContextFactory
java.naming.provider.url=iiop\://USER-PC\:2000
java.naming.security.principal=user
ant.project=default-jms-providers
java.naming.security.credentials=*
```

Examples This example logs you out of a session of the default JMS:

```
 jms-logout default
```

And indicates if the logout was successful:

```
Logged out.
```

See also jms-login

migrate

Description

Migrates J2EE applications, Web applications, Web services, EJBs, connectors, application clients, roles, configured queues, configured topics, queue connection factories, topic connection factories, and other entities from EAServer 5.*x* to EAServer 6.0.

The migrate command has these restrictions:

- You cannot migrate individual application clients because EAServer 5.*x* does not support export of individual application clients.
- You can migrate PowerBuilder packages, but not individual components. A package to be migrated cannot be of mixed component types.
- All connection caches are migrated, except ServletPersistenceCache. EAServer 6.0 does not support Oracle version 7 and earlier, so these connection caches are not migrated.
- EAServer migrates JAAS configuration files using this process:
  a Checks EAServer 5.*x* for a JAAS Configuration.

  b Creates an EntityCollection that contains a JAAS configuration file, exports, then extracts the contents to get a local copy of the 5.*x* JAAS configuration file.

  c Merges the contents of the EAServer 5.*x* JAAS configuration file with the EAServer 6.0 JAAS configuration file.

  d Sets the properties for a default SecurityDomain to a login method of JAAS, and sets the appropriate JAAS section from the EAServer 5.*x* settings. See Chapter 10, “Security Configuration Tasks,” in the *EAServer Security Administration and Programming Guide* for more information.
- You can migrate an EAServer 5.*x* role if it contains at least one authorized user, or one excluded user. Digital IDs are not supported (or migrated) to EAServer 6.0. Also, the default roles from 5.*x* (Admin Role, Debug Role, ThreadManager) are migrated only if you have modified them.

- You can migrate CORBA Java and C++ components. However, you can migrate only packages, not individual components. You cannot migrate packages with mixed J2EE and CORBA components.
- You cannot migrate to a remote EAServer 6.0 installation.

Syntax

migrate <*-options*> <*-deploy_options*>

Optionally, specify a timeout for interactions with the 5.*x* server:

```
migrate -Ddjc.migrateJagtoolTimeout=600 <-options> <-deploy_options>
```

A timeout prevents the migration tool from hanging in the rare occurrence that jagtool commands to the remote server encounter problems.

Option	Description
`-gui`	The default. If you use this option, all other options are ignored, and migrate starts a graphical user interface with which to migrate your EAServer 5.*x* entities. See the *Migration Guide* for complete instructions.
`-console`	Runs the migration tool from the command line and allows other command line options. If you do not enter the -console option, migrate uses the -gui option.
`-entity`	If you use this option, migrate allows you to migrate specific entities using the form *EntityType:EntityName*. For example: `migrate -entity ConnCache:Javacache`
`-eas5dir`	The EAServer 5.*x* installation directory from which you are migrating. You must either specify the full path to the EAServer 5.x directory or, if you do not specify a path, migrate uses the JAGUAR environment variable. If JAGUAR is empty, an error displays.
`-eas5outdir`	A temporary directory used by the migrate command. The default is a temporary directory in *${djc.home}/migrate*.
`-eas5host`	The host name on which the EAServer 5.*x* installation resides. The default is the value of the JAGUAR_HOST_NAME environment variable set in *bin/setenv.bat* (or *setenv.sh*) of the EAServer 5.*x* installation directory
`-eas5port`	The EAServer IIOP port number used to connect to EAServer 5.*x*. The default is 9000.
`-eas5user`	The name of the user connecting to EAServer 5.*x*. The default is jagadmin.
`-eas5password`	The password for *eas5user*. The default is an empty password (no value).
`-host`	The host name on which the EAServer 6.0 installation resides. The default value is the name of the host on which EAServer 6.0 resides.
`-port`	The EAServer IIOP port number used to connect to EAServer 6.0.
`-adminuser`	The name of the administrative user connecting to EAServer 6.0.
`-adminpwd`	The password for *adminuser*.

Option	Description
`-eas5httpport`	The EAServer HTTP port number used to connect to EAServer 5.*x*. The default is 8000. This is required for migrating EAServer 5.*x* Web services. See the *Migration Guide* for more information about migrating Web services.
`-httpport`	The EAServer port number used to connect to EAServer 6.0. This is required for migrating EAServer 5.*x* Web services.
`-doCheck true | false`	Examines the EAServer 5.x installation to determine which entities are available to migrate. The default is true.
`-doExport true | false`	Exports the entities to the migration output directory. The default is true.
`-doDeploy true | false`	Deploys the exported entities to the EAServer 6.0 installation. The default is true.
`-overwrite true | false`	Overwrite the contents of the output directory location. The default is false.
`-verbose`	Displays output messages of the migration command to the console.
`-quiet`	Migration output messages are minimal.

The migration is performed in three steps:

1. Check – examines the EAServer 5.*x* server to determine the entities available to migrate.
2. Export – the EAServer 5.*x* entities are exported to the migration output directory.
3. Deploy – exported entities are deployed individually to the destination EAServer 6.0 installation.

Note When overwrite is true at the "Check" step, all files in the migration output directory are deleted.

When overwrite is true at the "Deploy" step, deploying an entity fails if the entity already exists in the target EAServer 6.0 installation.

After running migrate, check the *migrate.log* file for error and informational messages. *migrate.log* is located in the EAServer 6.0 *bin* subdirectory.

Examples

Example 1 This example migrates an existing EAServer 5.*x* repository, identified by the JAGUAR environment variable, to the current EAServer 6.0 installation with overwrite set to true:

```
migrate -overwrite true
```

Example 2 This example migrates from a specific EAServer 5.*x* directory, using the doExport option, then uses doDeploy to deploy to an EAServer 6.0 installation:

```
migrate -eas5dir f:\sybase\jag52025p5 -eas5host
```

```
testhost -eas5port 9000 -eas5user jagadmin -doExport -
eas5outdir c:temp\migrate1

migrate -doDeploy -eas5outdir c:temp\migrate1

2005-09-16 14:08:52.210 INFO  main [MigrateTool]
Migrated archive files are in c:emp\migrate1
2005-09-16 14:08:52.835 INFO  main [DeployTool]
Extracting files from archive: C:
emp\migrate1\HelloWorldApp.ear
```

recompile

Description

Runs the recompile target in an entity's configuration file to apply changes to the entity properties and regenerate runtime classes required to integrate the entity implementation files into EAServer.

Syntax

recompile *entity* [*ant-opts*]

Option	Description
entity	The name of the entity to recompile, in the format described under "Entity references" on page 167
ant-opts	Any additional command line options to be passed to Ant

See also

configure, refresh

Chapter 2, "Ant-Based Configuration," in the *Automated Configuration Guide*

refresh

Description

Runs the refresh target in an entity's configuration file to reload the entity implementation classes and apply property changes that affect runtime behavior, such as JNDI name bindings or security constraints.

To change properties after modifying a configuration file, run configure or recompile, then run refresh.

Syntax

refresh *entity* [*ant-opts*]

Option	Description
entity	The name of the entity, in the format described under “Entity references” on page 167
ant-opts	Any additional command line options to be passed to Ant

See also

configure, recompile, restart-server

Chapter 2, “Ant-Based Configuration,” in the *Automated Configuration Guide*

restart-server

Description

Restarts EAServer. restart-server uses the wsh script to send a restart command to the server. Before you can use this script, you must first use the wlogin command to log in.

You can specify a server name, if you have previously called wlogin to log in to that server.

Syntax

restart-server [*host*:*port*]

Where ***host*:*port*** is the host name and HTTP port number of the server to restart. If not specified, the default is the name of the host where the command is run, and port 8000.

Note Instead of using the *host:port* syntax, you can use the -local parameter to restart any local servers that are running the “LocalRestart” scheduled task.

run-appclient

Description

Runs a J2EE application client.

Syntax

See “Running application clients” in Chapter 4, “Creating Application Clients,” in the *Enterprise JavaBeans User’s Guide*.

run-server

Description

Starts a server.

Syntax

run-server.bat [*server*] [-debug] [-jpdaSuspend] [-verbose]

Option	Description
server	The name of the server; the default is the name of the machine on which EAServer is installed.
-debug	Runs the server in debug mode.
-jpdaSuspend	If you are running the server in debug mode, this option suspends the server at start-up time, which allows you to set breakpoints in code that execute at start-up time. You can resume execution of the server with your Java debugger.
-verbose	Displays additional information useful for debugging.

Usage

run-server starts the same server in the same command or shell that you are using. To start a server in a new window, run start-server.

In a new installation, you must specify administration password with set-admin-password before starting the server.

See also

start-server, stop-server

service

Description

Installs EAServer as a Windows service and allows you to manage that service. You can create a Windows service with a particular service name. You can also specify which server to run in the service. You can create multiple services for multiple servers.

Syntax

service.bat [-servicename *-name*] [-*serviceOption*]

Option	Description
name	The name of the service. If not specified, the default service name is "EAServer."
-debug	Runs the service command in debug mode. Run the application from command line for debugging purposes. The service does not need to be installed. You can simulate and test service stop, pause, and continue behavior on the command line by starting the service with the -debug option. Use Ctrl+C for stop service, Ctrl+Break toggles between pause and continue service.

Option	Description
`-start`	Starts a service.
`-stop`	Stops a service.
`-install`	Installs a service.
`-restart`	Stops, then starts a service.
`-remove`	Removes a service.
`-config`	Reconfigures an already installed service. For example, if you make any changes to settings such as CLASSPATH, and you need the service to reflect the change, run the -config option to reconfigure the service settings stored in the Registry.
`-console`	Runs the service command in a console window.

Usage

service accepts a -servicename argument. If not specified, the default service name is “EAServer.”

service also accepts any arguments that can be passed to run-server. For example, use -server to specify a particular server to run as a service.

set-admin-password

Description

Interactively sets or resets the master administration password.

Syntax

set-admin-password

At the prompt, enter and reenter the new password.

Usage

This command is equivalent to running:

```
set-password admin@system
```

Passwords are case sensitive and must meet all these requirements:

- Be composed entirely of letters and the digits 0 – 9
- Contain at least 6 characters
- Contain at least one digit
- Contain at least two letters

See also

set-password

set-ant-opts

Description

Sets up the environment variable DJC_ANT_OPTS to hold various Java virtual machine options to send to Ant. You can include your own options by creating the file *set-ant-opts-local*. Just before Ant is invoked, ANT_OPTS=$DJC_ANT_OPTS is set.

Syntax

```
set-ant-opts
```

set-java-home

Description

Sets the directory on the local machine that contains the JDK.

Syntax

```
set-java-home
```

set-password

Description

Sets the password for a user.

Note Sybase recommends that you do not pass new passwords on the command line, particularly on UNIX systems, where another user can see the new password by running the ps command to view current operating system processes.

Syntax

set-password *user*[*@domain*] [*new-password*]

Option	Specifies
user	The user name.
@*domain*	The domain that the user name belongs to. If not specified, the default is "default".
new-password	The new password text. If not specified, set-password prompts you to enter and reenter the password.

Usage

Password rules are determined by properties of the security domain (see "Setting domain properties," in the *EAServer Security Administration and Programming Guide*), and can be changed by the system administrator. By default passwords are case sensitive and must meet all these requirements:

- Be composed entirely of letters and the digits 0 – 9

- Contain at least six characters
- Contain at least one digit
- Contain at least two letters

See also — set-admin-password

set-protected-property

Description — Sets the value for a property (usually a password) in protected storage.

Syntax — set-protected-property <*property*>@<*component*>:<*instance*> [<*prompt*>]

Option	Description	Required
property	The property that you are protecting	Yes
component	The component to which the property belongs	Yes
instance	The instance of the property	Yes
prompt	The prompt to enter the password displays after you enter the set-protected-property command	

Examples — This example clears the password history for the user guest that belongs to the test domain:

```
set-protected-property
password@com.sybase.djc.sql.DataSource:default "SQL
Password"
```

When you see this prompt, supply the password used to protect the property:

```
SQL Password:
```

See also — clear-protected-property

start-server

Description — Starts a server in a new window.

Syntax — Same as run-server.

Usage: start-server spawns a new command window (on Windows) or xterm window (on UNIX) and calls run-server to start the server in that window.

See also: run-server

stop-server

Description: Shuts down a server.

Syntax: stop-server [*host*:*port*]

Where ***host*:*port*** is the host name and HTTP port number of the server to log in to. If not specified, the default is the name of the host where the command is run and port 8000.

Note Instead of using the *host:port* syntax, you can use the -local parameter to stop any local servers that are running the "LocalStop" scheduled task.

You must have an active wlogin session with the server, using an account that has administrator role privileges on that server.

See also: run-server, wlogin

stub-compiler

Description: Generates and compiles EJB client stub classes.

Syntax: stub-compiler *interface1 interface2* ...

Where *interface1*, *interface2*, and so forth, are the names of EJB remote and home interfaces, for example:

```
stub-compiler com.sybase.easerver.tutorials.ejb.Query
com.sybase.easerver.tutorials.ejb.QueryHome
```

The code base to load the specified classes must be listed in the CLASSPATH environment variable.

stub-compiler compiles the stub implementation classes to the *genfiles/java/classes* subdirectory of the EAServer installation. Stub classes are in the Java package named by appending iiop_stubs to the package name of the interface class. For example, for interface com.sybase.easerver.tutorials.ejb.Query, the Java stub package is com.sybase.easerver.tutorials.ejb.iiop_stubs.

Usage

You can run stub-compiler to manually generate stubs for any EJB remote or home interface. You need not manually generate stubs for EJB-JAR files deployed to EAServer. When you deploy an EJB-JAR file, EAServer automatically creates stubs under the *genfiles/java/classes* subdirectory of the EAServer installation.

Use stub-compiler if the Java client will run with a JRE, rather than a full JDK. That is, if the client does not have a Java compiler available.

See also

idl-compiler

undeploy

Description

Removes an entity and deletes files associated with it.

Syntax

undeploy [-keepConfigFiles *true*|*false*] [-undeployListOnly] [-server *host*:*port*] *entities*

Option	Description
`-keepConfigFiles` *`true-false`*	Preserves the configuration files for removed entities, which are located in the *config* directory of the install.
`-undeployListOnly`	Lists files that would be removed and other actions that would be performed by running the command, but does not remove the specified entities.
`-server` *`host`*`:`*`port`*	Remotely deploys the archive to the server running on the specified host and listening on the specified HTTP port. If you do not specify a server, the archive deploys locally to the same installation where you run the deploy command. You must have an active wlogin session with the server, using an account that has administrator role privileges on that server. **Note** Instead of using the *host:port* syntax, you can pass the server name, if it is defined in the local Repository.

Option	Description
entities	The names of the entities to be removed, in the format described under "Entity references" on page 167. You can specify multiple entities, separated by commas. You can use wildcards to represent multiple entities. For example, use `ejbjar-foo*` to specify all EJB modules that begin with `foo`, or `appclient-*` to specify all application clients.

Usage

If an entity's Ant configuration file specifies additional files to include when exporting the entity, using the includeOtherFiles property, these files are not deleted when you undeploy the entity.

See also

deploy, export

wfs

Description

Issue a file service command to a server or cluster from the command line.

Syntax

wfs supports these command options:

wfs [- svr <server> | -cluster <cluster>] get *<remote-file> <local-file>*

wfs [- svr <server> | -cluster <cluster>] put *<local-file> <remote-file>*

wfs [- svr <server> | -cluster <cluster>] delete *<remote-file>*

Where ***`server`*** or ***`cluster`*** is the name of the server or cluster to which you are issuing the command.

You must log in to the server (host and port) to which you are issuing the command before running wsh. For example:

```
wlogin test-pc:8000
```

Command	Description
`get`	Gets a file <*remote-file*> from the server or cluster to which you are connected and creates a local file with the name <*local-file*>.
`put`	Puts a file <*local-file*> on the server or cluster to which you are connected with the name <*remote-file*>.
`delete` <*file*>	Deletes the file <*remote-file*> from the server or cluster to which you are connected.

Examples

From the *bin* subdirectory of the EAServer installation, this example logs in to the server test-pc, then deletes *HelloWorld.jar* from the *deploy* subdirectory:

```
wlogin test-pc:8000
Username: admin@system
Password: admin1
wfs delete ~/deploy/HelloWorld.jar
```

wlogin

Description

Log in to a server.

Syntax

wlogin [*host*:*port*]

Where ***host*:*port*** is the host name and HTTP port number of the server to log in to. If not specified, the default is the name of the host where the command is run and port 8000.

wlogin saves credentials with case-sensitive host information. When running deploy or other tools, you must specify the host name exactly as it was specified when creating the wlogin session.

Note Instead of using the *host:port* syntax, you can pass the server name, if it is defined in the local repository.

Usage

wlogin saves the credentials required to create an HTTP session on the specified server and establishes your credentials to run command line administration tools against that server.

Credentials are saved in individual files based on your operating system user name in the installation subdirectory *Repository/Instance/com/sybase/djc/security/DataProtection*. EAServer assigns read-write permission only for the current user, and encrypts the session data.

wlogin credentials remain valid until you call wlogout or remove the credential property file that matches your operating system user name. Even if you log out (or shut down) your computer, the stored credentials remain valid the next time you log in to the same computer as the same operating system user.

See also

wlogout

wlogout

Description

Log out from a server.

Syntax

wlogout [*host*:*port*]

Where ***host*:*port*** is the host name and HTTP port number of the server from which to log off. If not specified, the default is the name of the host where the command is run and port 8000.

Note Instead of using the *host:port* syntax, you can pass the server name, if it is defined in the local repository.

See also

wlogin

wsh

Description

Issue a command to a server or cluster from the command line.

Syntax

wsh [- svr <***server***> | -cluster <***cluster***>] <command>

Where `server` or cluster is the name of the server or cluster to which you are issuing the command. Executes <*command*> relative to *$djc.home/bin/<command>*.

You must log in to the server (host and port) to which you are issuing the command before running wsh. For example:

```
wlogin test-pc:8000
```

Command	Description
`login`	Logs in the user.
`logout`	Logs the user out.
`cat` <*file*>	Displays the contents of <*file*> to the console.
`ls` <*dir*>	Lists the contents of the directory <*dir*>.
`mkdir` <*dir*>	Creates the directory <*dir*>.
`rm` <*file*>	Deletes the file <*file*>.
`rmdir` <*dir*>	Deletes the directory <*dir*>.
`extract` <*jar*>	Extracts the contents of the JAR file <*jar*> to the absolute path of “/”.
`get-statistics`	Displays the current server statistics to the console.
`print-server`	Displays the current server boot cycle to the console.

Command	Description
`server-status`	Echoes the current server status to the console.
`refresh` <*entity*>	Refreshes the entity, specified as *entityType*:*entityName*.
`refresh-server`	Refreshes the server to which you are logged in.
`restart-server`	Restarts the server to which you are logged in.
`stop-server`	Stops the server to which you are logged in.
`start-module` <*name*>	Adds <*name*> to the server's startModules.
`stop-module` <*name*>	Removes <*name*> from the server's startModules.

Examples

From the *bin* subdirectory of the EAServer installation, this example logs in to the server test-pc, then lists the contents of the *deploy* subdirectory:

```
wlogin test-pc:8000
Username: admin@system
Password: admin1
wsh ls ~/deploy
```

CHAPTER 13 JNI Compiler

This chapter describes the Java Native Interface (JNI) compiler.

Overview

The JNI compiler is a command line tool that can generate:

- C++ proxies for accessing Java classes and interfaces, which are more productive and less error prone than direct JNI calls.
- C++ classes for implementing Java native methods, which are more efficient than IDL-based component calls from Java to C++ or vice versa.

You may want to use the JNI compiler:

- For developing C++ clients that talk to EJB components or JMS using the new EJB/JMS client runtime features (for example, data compression), which are not available in our CORBA C++ client runtime.
- When you need to call legacy C/C++ code from your components, and you would prefer to write EJBs with a few native methods than to write CORBA C++ components.

The JNI compiler works independently of CORBA IDL.

Using the JNI compiler

❖ **Running the JNI compiler**

1. Change to the EAServer *bin* directory.
2. Run the jnicc script, using the options described in Table 13-1.

```
jnicc.bat
```

Table 13-1: JNI compiler options

Option	Description
-verbose	Prints compiler actions at each line.
@options-file	Specify the name of a file that contains more options. You can also use this option within a file.
-cpp	Sets the target language to C++.
-class *class-name*	Specify a Java class name to generate a proxy in the target language for this class.
-interface *interface-name*	Specify the name of a Java interface to generate a proxy in the target language for this interface.
-field *class.field* -field *class.**	Specify a class name and optionally a field accessor method to include in the proxy for this class.
-method *class.method* -method *class.**	Specify a class name and optionally a method name to include in the proxy for this class.
-native *class-name*	Specify a class name to generate Java native methods that delegate to this target language class.
-o*base-name*	Set the base name for generated output files; the default is *java_classes*.
-D*xxx.yyy=value*	Set the Java system property *xxx.yyy* so it can be referenced as $(xxx.yyy).
-D*macro=value*	Include a `#define` macro for the C++ compiler.
-I*include-directory*	Specify the path to the *include* directory for the C++ compiler.
source-file.cpp	Include C++ source files to compile.
-dll	Run the target language compiler on the generated output files and generate a Windows DLL.

Generated files

The JNI compiler generates a C++ proxy class header file and source file for each Java class or interface that you compile; for example, if you compile A.B.MyClass, the following files are generated:

- *A_B_MyClass.hpp* – C++ proxy class header file.
- *A_B_MyClass.cpp* – C++ proxy class source file.

Sybase recommends that you do not edit these generated files, because any changes you make are overwritten when you re-run the compiler.

-native option

If you compile a class using the -native option; for example:

```
jnicc -native X.Y.MyClass
```

The JNI compiler generates the following files:

- *X_Y_MyClassNativeMethods.hpp* – C++ native methods class header file.
- *X_Y_MyClassNativeMethods.cpp* – C++ native methods class source file.

  To implement the native methods, edit *X_Y_MyClassNativeMethods.cpp*, and add the code to implement the methods. If you re-run the compiler, the new version of the source file has a *.cpp.new* suffix. If you add new methods to the Java class, then re-run the compiler, you may want to copy the newly-generated empty methods from *X_Y_MyClassNativeMethods.cpp.new* to *X_Y_MyClassNativeMethods.cpp.*
- An output file that includes all the header files using #include directives; for example, *java_classes.hpp*.
- An output file that includes all the source files using #include directives; for example, *java_classes.cpp*.

Generated classes

The JNI compiler generates four C++ class files for each Java class or interface that you compile; for example, if you compile A.B.MyClass, the following files are generated:

- A_B_MyClass – proxy class.
- A_B_MyClassArray – array proxy class.
- A_B_MyClassArrayElement – array element proxy class (for internal use).
- A_B_MyClassReturnValue – return value proxy class (for internal use).

If your C++ compiler supports namespaces, you can refer to proxy and array classes using scoped names; for example, A::B::MyClass.

Primitive type mappings

The JNI compiler maps Java primitive types to the JNI-equivalent C++ types:

Java type	JNI-equivalent C++ type
boolean	jboolean
byte	jbyte
char	jchar
double	jdouble
float	jfloat
int	jint
long	jlong
short	jshort

Arrays of Java primitive types are mapped to proxy classes in the *java_lang_Object.hpp* header file, as follows:

Java type	C++ type	C++ type with namespaces
boolean[]	java_booleanArray	java::booleanArray
byte[]	java_byteArray	java_byteArray
char[]	java_charArray	java_charArray
double[]	java_doubleArray	java_doubleArray
float[]	java_floatArray	java_floatArray
int[]	java_intArray	java_intArray
long[]	java_longArray	java_longArray
short[]	java_shortArray	java_shortArray

The Java null type is mapped to the java_null proxy class; java::null with namespaces.

Using proxy classes

This section describes how to use C++ proxy classes.

Declaring variables

To declare C++ proxy class variables, you can use:

- The C++ default constructor; for example, the following code lines each create an instance of the A_B_MyClass proxy class:

```
A_B_MyClass mcl;
A::B::MyClass mcl; // Using namespaces
```

```
// Using namespaces after introducing
// "using namespace A::B" within the current scope
MyClass mc1;
```

The default constructor sets the values of the class variables to null.

- The C++ copy constructor; for example, the following code lines each create a proxy class variable called *mc2*, which is a copy of *mc1*:

  ```
  A_B_MyClass mc2(mc1);
  A_B_MyClass mc2(mc1.getMyObject());
  ```

  The values of *mc2* variables are set to the values of *mc1* variables.

Assigning variables

To assign proxy class variables, you can use any of the following statements, where *mc1* and *mc2* are both instances of the same proxy class.

- From another object:

  ```
  mc2 = mc1
  ```

- From an expression:

  ```
  mc2 = mc1.getMyobject();
  ```

- To set the value to null:

  ```
  mc2 = java_null
  ```

Comparing references

You can compare proxy class references to check for null or for equality.

- Check whether the value of the proxy class variable is null:

  ```
  if (mc1 == java_null)
  ```

- Check whether *mc1* and *mc2* refer to the same Java object:

  ```
  if (mc1 == mc2)
  ```

Calling constructors

Java constructors are mapped to C++ global functions whose names are created by adding the prefix "new" to the name of the constructor. For example, if Java class A.B.MyClass has a default constructor, you can call the C++ proxy class method as follows:

```
mc1 = new_A_B_MyClass();
mc1 = A::B::new_MyClass(); // when using namespaces
```

If Java class A.B.MyClass has a constructor that takes an int parameter, you can call the C++ proxy class method as in this example:

```
mc1 = new_A_B_MyClass(123);
```

Defining macros

If your C++ compiler does not support namespaces, you might want to define a few macros for commonly accessed classes and methods; for example:

```
#define null java_null
#define String java_lang_String
#define new_String new_java_lang_String
```

Array proxy classes

For all referenced Java array types, an array proxy class is generated, which contains:

- For all array types, a constructor that takes the array size as a parameter; for example, the following line creates an instance of A_B_MyClassArray, with a size of three:

  ```
  A_B_MyClassArray mca = new_A_B_MyClassArray(3);
  ```

- For arrays of primitive Java types only, a constructor that takes an array size parameter and initial values.

  ```
  jbyte values[] = { 0, 1, 2 };
  java_byteArray jba = new_java_byteArray(3, &values);
  ```

Array proxy classes overload the C++ array indexing operator, which is zero-based, as in Java.

- To get the second element of an array:

  ```
  jbyte a1 = jba[1];
  ```

- To set the third element of an array with the value from the second element:

  ```
  jba[2] = jba[1];
  ```

The String class

As with all other Java object types, java.lang.String is mapped to a proxy class, java_lang_String. This proxy class has a few special features:

- A C++ constructor that takes a char* parameter (assumes UTF-8 encoding):

  ```
  String s1("abc");
  ```

- An overloaded assignment operator, which allows you to assign a string in any of the following ways:

  ```
  String s2 = "xyz";
  String s3 = String("xyz");
  String s4 = new_String("xyz");
  ```

- A str method that returns a char* (UTF-8 format). The return value is automatically freed when the String from which it was obtained goes out of scope. The following code segment prints “xyz” to standard output:

  ```
  String s2 = "xyz";
  char* s2Str = s2.str();
  cout << s2.str();
  ```

- A strcpy method that returns a char* (UTF-8 format). The return value is a copy, which is not automatically freed. You must free it explicitly using the delete[] operator. The following code segment prints “xyz” to standard output, then frees the memory associated with *s2Str*.

  ```
  String s2 = "xyz";
  char* s2Str = s2.strcpy();
  cout << s2Str;
  delete[] s2Str;
  ```

To pass a String constant in a parameter of type Object, you can use an explicit String constructor, as illustrated in the following code segment:

```
java_util_HashMap map = new_java_util_HashMap();
map.put(String("abc"), String("xyz"));
```

Accessing Java fields and methods

The JNI compiler maps each Java instance field to a pair of methods in the C++ proxy class; for example, the Java field xyz is mapped to the getXyz and setXyz methods. The compiler maps final fields to a single method of the same name; field xyz is mapped to method xyz, which is a getter method only. If a final field is a primitive type, it is mapped to a C++ constant with the same name.

The following code fragments display how you can get field values using C++ proxy class methods:

```
jint i = mc.getMyInt();
String s1 = mc.getMyString();

// static field uses static method
String s2 = A_B_MyClass::getMyStaticString();

// static primitive field uses C++ constant
double pi = A_B_MyClass::PI;
```

The code fragments below illustrate how you can set field values using C++ proxy class methods:

```
mc.setMyInt(123);
mc.setMyStringField(String("abc"));

// static field uses static method
A_B_MyClass::setMyStaticString(String("xyz"));
```

To get the java.lang.Class definition for a class—which is equivalent to accessing a static field named “class” in Java—use the following syntax:

```
java_lang_Class myClass = A_B_MyClass::_class();
```

Java methods

Each Java method maps directly to a C++ method. The following code fragments illustrate how to call a method in a C++ proxy class:

```
mc.myVoidMethod();
String s1 = mc.myStringMethod(123);

// static field uses static method
String s2 = A_B_MyClass::myStaticStringField();
```

Intermediate values

To prevent memory leaks with JNI GlobalRefs, there are restrictions for using method return values. To call a method on a return value, the return value must be explicitly converted to the expected type. In the following example, the first line is valid because it converts the return value from mc.myStringMethod to a String before calling length(). The second line is not valid:

```
jint len = String(mc.myStringMethod()).length();
jint len = mc.myStringMethod().length(); // invalid
```

Java exceptions

Java exceptions are mapped to C++ proxy classes, so you can use normal C++ exception handling.

Catching Java exceptions

The following code sample illustrates how to catch a checked exception:

```
try
{
   mc.myBadMethod(456);
}
catch (A_B_MyException& ex)
{
   ex.printStackTrace();
}
```

Since the full set of possible unchecked exceptions (subclasses of java.lang.RuntimeException) cannot be known in advance, you can expect to catch an exception type only if the type's class name was passed to the JNI compiler using the -class option—see Table 13-1 on page 204.

The following code sample catches checked and unchecked exceptions, errors, and throwables:

```
try
{
   String s = java_null;
   jint n = s.length();
}
catch (my_special_RuntimeException& ex1)
{
   // Catches exceptions only if jnicc options
   // included: -class my.special.RuntimeException
}
catch (java_lang_RuntimeException& ex2)
{
   // Catches all unchecked exceptions
}
catch (java_lang_Exception& ex3)
{
   // Catches all checked and unchecked exceptions
}
catch (java_lang_Error& ex4)
{
   // Catches all errors
}
catch (java_lang_Throwable& ex5)
{
   // Catches all throwables
}
```

Throwing Java exceptions

Throwing Java exceptions is similar to throwing C++ exceptions.

To throw a new exception, use the syntax in either of the first two lines:

```
throw new_A_B_MyException("MyMessage");
throw new_A_B_MyException();

// Invalid because of space between "new"
// and A_B_MyException
throw new A_B_MyException();
```

To throw an existing exception object:

```
A_B_MyException ex = mc.getMyException();
throw ex;
```

Implementing native methods

To implement native methods:

1. Generate default (template) implementation classes using the -native compiler option. This generates C++ header and source files with a "NativeMethods" suffix. For example, if you compile A.B.MyClass, the generated source file is *A_B_MyClassNativeMethods.cpp*.
2. Edit the generated source file, in this case *A_B_MyClassNativeMethods.cpp*, and add the code to implement the methods.
3. Compile your source files into a DLL or shared library. On Windows, the easiest way to do this is to use the -dll option on the jnicc command line. On other systems, invoke your compiler separately afterward.
4. Load the DLL or shared library from your Java class with a static initializer by calling com.sybase.jni.SystemUtil.loadLibrary, as in the following example. Native methods are registered automatically.

```
public class MyClass
{
   static
   {
      com.sybase.jni.SystemUtil.loadLibrary("MyClass");
   }

   public native void myMethod();
}
```

5. To register native methods statically in C++ code, use the following syntax:

```
A_B_MyClassNativeMethods::__register();
```

You can find a sample implementation in the EAServer *samples/SimpleRowSet* directory.

Implementing private data

If the C++ class implementing your native methods requires access to C++ instance data, perform the following:

1 Add a private long type variable called "data" to your Java class:

```
public class MyClass
{
   static
   {
      com.sybase.jni.SystemUtil.loadLibrary("MyClass");
   }

   private long data;

   public native void myMethod();
}
```

2 When you run jnicc, it now generates the C++ source and header files *A_B_MyClassPrivateData.cpp* and *A_B_MyClassPrivateData.hpp*.Add all private instance data to the A_B_MyClassPrivateData class. You can add any methods you want to this class.

3 Add a public finalize method that calls a private destroy method. The JNI compiler ensures that when the destroy method is called, the C++ instance data is deleted.

```
public class MyClass
{
   static
   {
      com.sybase.jni.SystemUtil.loadLibrary("MyClass");
   }

   private long data;

   public native void myMethod();

   public void finalize()
   {
      destroy();
   }
```

```
        private native void destroy();
    }
```

4 Access the instance data from your native methods using the *data* variable—see the sample in *samples/SimpleRowSet*.

C++ compile-time options

If your C++ compiler supports namespaces and you want to use them to reference generated proxy classes, define the JNICC_NAMESPACE macro (with any value).

All generated code should be compatible with any C++ compiler that supports the standard JNI API. If issues arise with your C++ compiler, contact Sybase Technical Support.

CHAPTER 14 Systems Management

This version of Systems Management is based on the Java Management Extensions (JMX) agent management framework. It allows developers to monitor and manage EAServer applications, components, and the server itself.

Overview

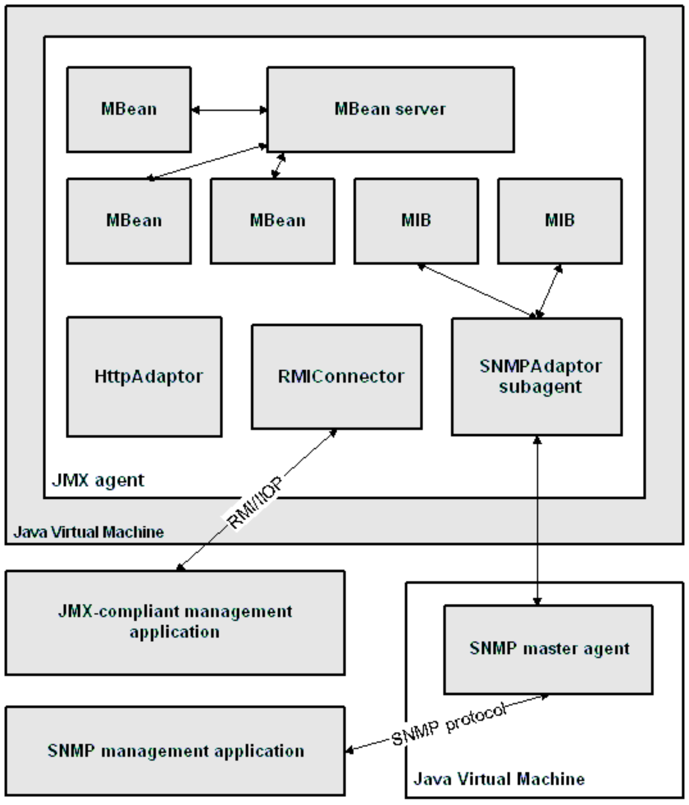
Systems Management exposes management and monitoring functionalities to other applications, such as the Simple Network Management Protocol (SNMP) and JMX-compliant management applications.

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.

Systems Management uses the MX4J implementation of the JMX version 1.2 API. 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 an 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. 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, except for a small number of methods that are appropriate only for local access to the MBean server.

Figure 14-1: Systems Management architecture



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 1.2.
- **SNMP support** Support for SNMP includes an SNMP master agent from J.AgentX.

  The master agent returns information contained in a management information base (MIB). You can view the SNMP MIBs using an SNMP management application.
- **SSL support** You can use SSL when connecting to the JMX agent.

JMX agent

Once started, the JMX agent consists of a JVM running in the same process as EAServer. 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 loads the MBeans and services that are defined in *DJC_HOME/config/jmx/boot.xml*.

The JMX agent is based on the MX4J Release 3.0.1.

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 life cycle 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, 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. These MBeans provide a view of remote processes, 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 log file for the process.

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; 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 adapters, HTTPAdaptor and SNMPAdaptor. You can use an HTTP client utility to connect to the HTTPAdaptor to process HTTP requests. The SNMPAdaptor allows you to use an SNMP management console to view SNMP information that is exposed via the deployed MIBs; NETWORK-SERVICES and J2EE. The SNMPAdaptor is an SNMP subagent—see “Connecting to the JMX agent” on page 226 for details about how to write clients that connect to the agent. An RMI connector supports RMI access to the server.

SNMP functionality

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

EAServer 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 protocol SNMPv1.

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 *start-master-agent.sh* or *start-master-agent.bat*, located in *DJC_HOME/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 *DJC_HOME/config/jmx* 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 *DJC_HOME/bin* directory, and enter:

  On UNIX or in an MKS shell on Windows – `start-master-agent.sh`

  On Windows – `start-master-agent.bat`

  start-master-agent writes output to *logs/master.stdout*.

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.

❖ Running the SNMP master agent as a Windows service

- Run the following command in the EAServer *bin* subdirectory, using the options described in Table 14-1.

  ```
  start-master-agent
  ```

  When the master agent is installed as a Windows service, the display name is "SybMaster."

Table 14-1: SNMP master agent service options

Option	Description
-h	Display the list of options available.
-start	Starts the master agent as a Windows service.
-stop	Stops the master agent service.
-restart	Stops, then starts the master agent service.
-install	Installs the master agent as the SybMaster Windows service.
-remove	Removes the master agent from the list of Windows services.

❖ 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—displays 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.

EAServer 6.0 supports all the MIBs that are defined in the J2EE MIB, including:

- **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. The j2eeMoGroup includes:
  - j2eeDomTable – domains.
  - j2eeSrvTable – servers.
  - j2eeJVMTable – Java virtual machine.
  - j2eeAppTable – applications.
  - j2eeModTable – modules in all applications.
  - j2eeBeanTable – EJBs in all applications.
  - j2eeSleTable – servlets in all applications.
  - j2eeAdapTable – resource adaptors in all applications.
  - j2eeRsrcTable – resources used by a server.
  - j2eeJCATable – JCA-managed connection factories.
  - j2eeJDBCTable – JDBC drivers and data sources used by a server.
- **j2eeStatistics** The following j2eeStatistics groups provide statistics for performance monitoring:
  - j2eeServletStatGroup – servlet statistics.
  - j2eeEjbStatGroup – EJB statistics.
  - j2eeJavaMailStatGroup – JavaMail statistics.
  - j2eeJtaStatStatGroup – JTA resource statistics.
  - j2eeJcaStatStatGroup – statistics of nonpooled connections and the connection pools associated with the referencing JCA resource or connection factory.
  - j2eeJdbcStatStatGroup – statistics of nonpooled connections and the connection pools associated with the referencing JDBC resource or connection factory.
  - j2eeJmsStatStatGroup – JVM statistics.

SNMP MIB implementation details

When an 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.

Because an SNMP client can generate many requests in rapid succession, it is not reasonable to generate a corresponding response 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.

Running the SNMP subagent

Before you run the SNMP subagent, verify that the JAVA_HOME environment variable points to a JRE 1.4 installation, and that the SNMP master agent is running so that the SNMP subagent can connect to it—see “SNMP master agent” on page 219.

You can configure the SNMP subagent to start automatically when EAServer starts, but you must ensure that the SNMP master agent starts first. To set up the SNMP subagent to start automatically, select SnmpService on the EAServer Start Components tab—see “Services tab” on page 38. The SNMP subagent is implemented by SNMPAdaptor.

To set the JMX logging level, see “Log/Trace tab” on page 30.

jmx.admin role is required to run SNMP subagent To start or stop the SNMP subagent, you must be a member of the jmx.admin role. When you run the commands to start or stop the SNMP subagent, the system prompts you to enter your user name and password, then validates your role membership.

❖ Starting the SNMP subagent

- Change to the *bin* directory.

  On Windows, run:

  ```
  start-agent.bat [-S <host>] [-P <RMI_port>] [-h]
  ```

  On UNIX, run:

  ```
  start-agent.sh [-S <host>] [-P <RMI_port>] [-h]
  ```

  where the command line options are:

Option	Description
-S *host*	Specifies the EAServer host name.
-P *RMI_port*	Specifies the RMI registry service port number.
-h	Prints the list of start-up options.

❖ Stopping the SNMP subagent

- Change to the *bin* directory.

  On Windows, run:

  ```
  stop-agent.bat [-S <host>] [-P <RMI_port>] [-h]
  ```

  On UNIX, run:

  ```
  stop-agent.sh [-S <host>] [-P <RMI_port>] [-h]
  ```

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.MBeanServerConnection 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.

Creating services

A service is an object that can run either within or outside of the JMX agent's VM. 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 217. To interact with external services, create service MBeans.

Creating service MBeans

Service MBeans must expose a well-defined life cycle API that provides the ability to start, stop, and check the status of managed services.

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 14-2.

Table 14-2: MBean required methods

Method	Description
void start()	Starts the MBean.
void stop()	Stops the MBean.

Method	Description
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.djc.management.jmx.services.ServiceSuport class is available to extend, if required. This class provides all the API methods described in Table 14-2 and Table 14-3. If you extend this class, you can override the methods defined in Table 14-3, and let the ServiceSupport class handle state management for you. The ServiceSupport class provides default implementations for all the methods listed in Table 14-3, so you can implement only the ones you need. The ServiceSupport class is in *DJC_HOME/lib/eas-server.jar*.

Table 14-3: 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

Index