SYBASE®

Troubleshooting Guide

# EAServer

6.0

DOCUMENT ID: DC10113-01-0600-02

LAST REVISED: June 2008



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

# Contents

# About This Book

**About This Book**

This book contains procedures for troubleshooting problems that EAServer users may encounter. The problems addressed here are those that the Sybase® Technical Support staff hear about most often. This guide is applicable to EAServer version 6.0, and its purpose is to provide:

- Information about common errors so you can resolve problems without help from Technical Support.
- A list of information that you can gather before calling Technical Support so they may be able to help resolve your problem more quickly.
- A greater understanding of EAServer.

**Audience**

This book is for EAServer administrators, and those responsible for administering, supporting, developing, or deploying client applications or components that run on EAServer.

**How to use this book**

This book includes these chapters:

- Chapter 1, “Monitoring Techniques,” introduces the tools and techniques available for logging events and errors, and monitoring the server.
- Chapter 2, “Common Problem Areas,” describes common problem areas such as start-up and connection problems, server crashes and hangs, and configuration issues.
- Chapter 3, “Performance Issues,” provides an overview of resources available for performance tuning and troubleshooting.
- Chapter 4, “Exception Handling,” explains how to handle errors in EAServer components and applications.
- Chapter 5, “Common Error Messages,” contains a listing of server errors with links to additional information.
- Chapter 6, “Advanced Topics,” surveys more advanced topics like debuggers, stack traces, and tools for memory management and runtime monitoring.

**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 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 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* 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 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* (this book) describes procedures for troubleshooting problems that EAServer users may encounter. This document is available on the SyBooks Online Web site at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc10113_0600/html/eastg/title.htm.

**jConnect for JDBC documents** EAServer includes the jConnect™ 6.0.5 driver to allow JDBC access to Sybase database servers and gateways. The *jConnect 6.0.5 Programmer's Reference* is available on the SyBooks Online Web site at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc39001_0605/html/prjdbc/title.htm.

**Sybase Software Asset Management User 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 SyBooks Online Web site at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc00530_0200/html/sysamug/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:<br>• Command names used in descriptive text<br>• C++ and Java method or class names used in descriptive text<br>• Java package names used in descriptive text<br>• Property names in the raw format, as when using Ant or jagtool to configure applications rather than the Management Console |
| *variable*, *package*, or *component* | Italic font indicates:<br>• Program variables, such as *myCounter*<br>• Parts of input text that must be substituted, for example:<br>`Server.log`<br>• File names<br>• 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:<br>• Information that you enter in the Management Console, a command line, or as program text<br>• Example program fragments<br>• Example output fragments |

## Other sources of information

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

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

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

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

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

**Sybase certifications on the Web**

Technical documentation at the Sybase Web site is updated frequently.

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

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click Certification Report.
3. In the Certification Report filter select a product, platform, and timeframe and then click Go.
4. Click a Certification Report title to display the report.

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

1. Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.
2. Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.

3 Select Search to display the availability and certification report for the selection.

❖ **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 Web console supports working without a mouse. For more information, see "Keyboard navigation" in Chapter 2, "Management Console Overview," in the *EAServer System Administration Guide*.

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.

**If you need help**

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

CHAPTER 1

# Monitoring Techniques



## Overview

This chapter introduces several useful tools and techniques for monitoring and analyzing your EAServer applications and environment.

Some of the resources described in this chapter serve more than one purpose. For example, a tool may provide both monitoring and analysis features, and therefore is mentioned in multiple sections.

**Note** Some techniques and suggestions in this document require that you restart the server for changes to take effect, or to test the result and impact. This is noted when required.

## Gathering information

When you are investigating an EAServer error, gather the following information to convey to Sybase Technical Support; it may help them expedite a solution:

1 Full text of error message and crash details, as appropriate.

2 Information about your EAServer installation:

- Version number
- Build number
- Edition

This information appears at the start of the server log. See “Logging and statistics” on page 4 for more information about log files.

3 Information about the EAServer process that is running in your environment, including version numbers of the DLLs or libraries loaded—see “ListDLLs” on page 69. To list which libraries are loaded and their load locations, use pmap, or a similar command for your platform.

4 Relevant log files, located in the *logs* subdirectory:

- **Server log** The server log may contain:
  - Console output
  - DriverManager messages
  - Exception cross-references
  - Transaction cross-references
  - Application exceptions
  - System exceptions
  - Remote IIOP and RMI-IIOP method invocations
  - Local RMI method invocations
  - EJB component invocations
  - Web component invocations
  - Java Message Service (JMS) operations
  - JDBC driver activity
- **HTTP request log** HTTP request logs are named *serverName-http-YYYY-MM-DD.log*, where *serverName* is the name of the server, and *YYYY-MM-DD* is the current date.

5 Information about your platform:

- Number of CPUs
- Operating system version and patch levels
- Memory

6 Environment variables, many of which are defined in the setup script *djc-setenv.sh* (UNIX) or *djc-setenv.bat* (Windows):

- DJC_HOME
- JAGUAR_HOST_NAME
- CLASSPATH
- LD_LIBRARY_PATH
- DJC_JDK_14 and DJC_JDK_15

Not all variables are defined in the setup script. The server and tools start-up scripts configure PATH, CLASSPATH, BOOTCLASSPATH, and other settings.

Also check the user-defined scripts *local-setenv.sh* (UNIX) or *local-setenv.bat* (Windows) for additional settings.

See “Configuration issues” on page 34 for more information on environment variables.

7 Information about the server JDK; including:

- Java version (from the server log file)
- VM type (from the server log file)
- Other server properties that configure the Java VM

8 Database:

- Database server
- Database version/driver
- Connection type (ODBC, JDBC, native)

9 Application information:

- Nature of the application (for example, an order entry system)
- Application type: client/server or Web application
- Number of concurrent users that access the application during peak time
- Maximum amount of data that the components retrieve
- Amount of data returned to the clients
- If service components exist, their function (verify that they are not transactional)

- If shared components (versus multiple instance components) exist, their purpose
- Whether components are being pooled
- If stateful components exist, their function; also, whether SetComplete and SetAbort functions are called for these components
- Types of components in use: PowerBuilder®, Web applications, servlets, JSPs, and so on
- For applications that include PowerBuilder components, the number of:
  - PowerBuilder components used in the application
  - Whether any PowerBuilder components are invoked by a non-PowerBuilder client; and if so, the type of client
  - Whether the SetComplete and SetAbort functions are being called for stateful components, or whether the auto-demarcation deactivation component property is set for the components

# Logging and statistics

Log files and statistics provide useful troubleshooting information.

## EAServer log

This log file contains the server version, build number, listener addresses, Java virtual machine (VM) version and type, trace and debug messages, all other messages logged by the server, and component output.

Use any of these techniques to output messages to the EAServer log file from an EAServer component or application:

- Enable debug, trace, or other properties as outlined in “EAServer tracing” on page 9.
- Call System.out.print from a Java component.
- Call jaguar.server.Jaguar.writeLog method in a Java component. See the com.sybase.jaguar.server.Jaguar class documentation in Chapter 1, “Java Classes and Interfaces,” in the *EAServer API Reference Manual*.

- Call JagLog from a C or C++ component. See JagLog in Chapter 2, “C Routines Reference,” in the *EAServer API Reference Manual*.
- Utilize the ErrorLogging object for a PowerBuilder component. See “Error logging service” in the *PowerBuilder Application Techniques* manual for usage information.

The server writes messages to the <*serverName*>.*log* file.

The server may exit immediately without logging errors if there are errors in the server configuration; in this case, the server prints error messages to the shell window (console) where it was started.

## Integrating with other logging systems

EAServer includes a configurable logging mechanism that allows integration with the JDK 1.4 or JDK 1.5 Java logging package, or the Apache Log4j logging system. A server’s logging properties are defined in a **log profile**, which defines the logging subsystem used, as well as other properties, such as output destinations, formats, and the level of severity required before a message is recorded.

You can use the following logging subsystems:

- The built-in EAS subsystem, which provides:
  - The ability to configure log levels so that messages below a specified level of severity are discarded
  - Optional archiving and compression of previous log file versions
- Apache Log4j, which is commonly used on large projects. For more information, see the Apache Log4j Documentation at http://jakarta.apache.org/log4j/docs/api/overview-summary.html.
- The Java Logging package, included in JDK versions 1.4 and 1.5. This API is the Sun-proposed standard for logging in Java applications. For more information, see the Java Logging documentation at http://java.sun.com/j2se/1.4.1/docs/guide/util/logging/overview.html. To use this package, your server must be running JDK 1.4 or a later JDK version.

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

#### Logging APIs

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

- System.out.println or Jaguar.writeLog from Java code running in the server
- ErrorLogging.log from PowerBuilder NVO (nonvisual object) components
- JagLog from C or C++ components

In addition, if you use Log4j or the Java Logging system, you can log messages from in-server Java code by calling the logging API directly.

#### Managing system logging

"Configuring system logging" in Chapter 3, "Creating and Configuring Servers," in the *EAServer System Administration Guide* describes how to manage system logging.

### Component logging

To enable tracing for all the components that have trace flags available under the com.sybase.djc package, start EAServer using:

```
-Ddjc.trace=com.sybase.djc.*
```

**Note** Component logging can degrade server performance; therefore, Sybase suggests that you use it only for short durations, and turn it off as soon as you obtain the necessary information.

## HTTP statistics

Use the Management Console to view the HTTP statistics for a server.

| Statistic | Description |
|---|---|
| Accepted Connections | The number of HTTP connection requests 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 open HTTP connections |
| Average Duration of Requests | The average duration (in milliseconds) of HTTP requests |
| Number of Requests | The total number of HTTP requests |

You can enable or disable HTTP request logging on the server's Log/Trace tab.

## IIOP statistics

To trace IIOP traffic between EAServer and an IIOP client, use the rmiiopTrace flag when you start the server.

Use the Management Console to view IIOP statistics for a server.

| Statistic | Description |
|---|---|
| Accepted Connections | The number of IIOP connection requests accepted by the server |
| Open Connections | The number of open IIOP connections |

You can enable or disable IIOP session activity on the server's Log/Trace tab.

IIOP statistics may be helpful in diagnosing:

- Client application issues, such as login failures
- Component issues, including:
  - Method invocation errors
  - Result sets not returned as expected
  - Trouble connecting to the target database

- Intercomponent call errors

The log may be difficult to decipher when multiple clients are talking to the server in parallel. If possible, run only one client when using the IIOP log information.

**IIOP logging is verbose** IIOP logging can quickly fill up the disk and degrade server performance. Use IIOP logging only for short durations; turn it off as soon as you gather the information you need.

## Other useful data

Other useful data comes from:

- Server console messages, which may provide additional error information.
- Licensing output in the log file.
- Client-side log files. See “EAServer tracing” on page 9 for more information.

# Using stack traces

Stack traces, dump files, and core files contain useful information about what a server process is doing at a given time, such as:

- Methods called
- Memory information
- Active thread information

These files are time consuming to read and not always easy to understand. Try simple troubleshooting techniques first. Use the server log first, which is much more readable, to review server and component output and check any errors raised. Next, check the dump file to get an idea of which debug/trace flags should be turned on. This may help identify things like operating system signal issues.

For information about how to obtain various trace files for troubleshooting, see “Stack traces, dump files, and core files” on page 74.

**Note** Stack traces, dump files, and core files are not mutually exclusive. Depending on the platform and tools or options you use, the output file may contain one or more types of detail.

## Obtaining stack traces

The EAServer process stack trace contains information about the active threads for the process.

❖ **Obtaining a stack trace in UNIX**

- In UNIX, to generate a JVM full thread dump, enter:

  ```
  kill -QUIT EAServer_process_id
  ```

  This does not work if you use “-Xrs” as a JVM parameter.

  **Note** AIX may need special configuration for a full dump.

  On the Solaris platform, to generate a C back trace, you can use pstack, which is located in */usr/proc/bin*.

Another alternative is to generate a core file—see “UNIX core files” on page 75.

## EAServer tracing

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:

1 Use the Management Console to connect to EAServer.

2 Expand the Servers folder, and select the server.

3 On the Log/Trace tab, select or unselect the following properties:

- **Capture Console Output in Log** Writes messages that display in the console to the server log file.

- **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.
- **Enable java.sql.DriverManager Log** Writes DriverManager log messages to the server log.
- **Generate Exception Cross-Reference** Cross-references exceptions in the server log file.
- **Generate Transaction Cross-Reference** Cross-references transactions in the server log file. To start transaction tracing from the command line, use the logTransaction flag when you start EAServer
- **Log Application Exceptions** Writes application exceptions to the server log. If a deployed module requires that application exceptions be logged, set 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** Writes system exceptions to the server log. To log system exceptions, use the same technique described in Log Application Exceptions, above.
- **JMX Logging Level** The logging level for the Systems Management JMX agent. See "Running the SNMP subagent" in Chapter 14, "Systems Management," in the *EAServer System Administration Guide*.
- **Enable RMI-IIOP Trace** Traces remote method invocations for IIOP and RMI-IIOP in the server log. You can also enable RMI-IIOP tracing from the command line, which does not change the value in the server properties file. To start IIOP tracing from the command line, use the rmiiopTrace flag when you start the server.

  **Warning!** Enabling RMI-IIOP trace 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. To start tracing from the command line, use the rmiiopTraceLocal flag when you start the server.

- **Enable EJB Trace** Traces EJB component invocations in the server log. To start EJB tracing from the command line, use the ejbTrace flag when you start EAServer.
- **Enable Web Trace** Traces Web component invocations in the server log.
- **Enable JMS Trace** Traces Java Message Service (JMS) operations in the server log. All public and protected JMS provider methods are traced. To start JMS tracing from the command line, use the jmsTrace flag when you start EAServer.
- **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. To start SQL tracing from the command line, use the sqlTrace flag when you start EAServer.

You can also enable these logging and tracing properties from the command line or in the *setenv.bat[sh]* file—see "Managing system logging" on page 6.

For information on viewing the log files, see "Viewing server log files" in Chapter 11, "Runtime Monitoring," in the *EAServer System Administration Guide*.

## Java exception traces

There are two types of stack traces, thread-level and process-level. When you catch an exception in your component, you can print the thread-level stack trace using the printStackTrace method in the component's try/catch block:

```
try {
// whatever
}
catch (Exception e) {
{
// Important to avoid empty catch blocks
// Remember to output error information
   e.printStackTrace();
}
```

Many APIs such as JDBC and JNDI throw generic exceptions, with the original error information embedded as a nested exception. For JDBC, you can retrieve the nested exception using the java.sql.SQLException.getNextException method. Record the nested exception message and stack trace.

You may also need to print naming exceptions. For example, when creating proxies in an EJB client, a naming exception may be thrown. Since the lookup method must throw a NamingException, other errors can be embedded, such as the NamingException root cause. Often, you must retrieve details about the embedded error before you can diagnose the problem. The code below demonstrates how to retrieve and print the root cause information for a JNDI NamingException:

```
catch (NamingException ne)
{
   System.out.println("Naming exception: " +
                      ne.toString());
   ne.printStackTrace();
   throwable rc = ne.getRootCause();
   if (rc != null)
   {
     System.out.println("\nRoot cause: + rc.toString();
     rc.printStackTrace();
   }
}
```

If you do not catch exceptions in component code, EAServer catches the exception, logs the error, and aborts method execution. However, the server may not record all information required to debug the problem, such as nested exceptions. For this reason, catch exceptions in your own code and log the information required to diagnose the problem.

If serious internal errors occur, the Java VM may abort and produce a process-level stack trace that contains trace information for all Java threads, signals received, and thread states at the time of the error.

# Tracing network problems

A number of tools and techniques are available to monitor the operation of the server, its environment, and applications. Some useful network and protocol-level monitoring tools and resources are described below.

## TCP/IP

Network tracing tools for the TCP/IP layer include:

- netstat – displays current TCP/IP network connections and protocol statistics.
- TDImon – an application that lets you monitor TCP and UDP (User Datagram Protocol) activity, track network-related configuration problems, and analyze application network usage. This tool is available at the Sysinternals Web site at http://www.sysinternals.com.
- ping – attempts to elicit a response from a specified host.
- traceroute – helps you determine the route IP packets take to a network host.

## IIOP

Network tracing tools for IIOP include:

- Management Console monitoring—see “IIOP statistics” on page 7.
- IIOP output logging, which traces remote method invocations for IIOP and RMI-IIOP in the server log. See “EAServer tracing” on page 9.

## HTTP

Network tracing tools for HTTP include:

- Management Console monitoring—see “HTTP statistics” on page 6.
- HTTP request log—see “EAServer tracing” on page 9.

CHAPTER 2

# Common Problem Areas



## Installation issues

If you encounter installation issues, consult the *EAServer Installation Guide* for your platform, which can help you resolve problems and answer questions about:

- Installing
- Upgrading
- Licensing
- Adding components
- Reinstalling

To locate the *EAServer Installation Guide*:

1 Go to the Product Manuals site at http://infocenter.sybase.com.

2 Select EAServer 6.0, then select the *Installation Guide* for your platform.

## Migrating from earlier EAServer versions

To migrate EAServer entities from an earlier version to version 6.0, begin with EAServer version 5.5 or later, and run the EAServer 6.0 migration tool. To migrate from an EAServer version earlier than 5.5, first migrate to version 5.5, then run the 6.0 migration tool.

See the EAServer Migration Guide at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc00485_0600/html/easmig/title.htm.

# Server crashes, hangs, or disappears

This section describes techniques for isolating the cause if a server crashes, hangs, or disappears. These are guidelines only, subject to your application or environment; they may or may not help isolate the cause.

## Server crashes

A server failure (crash) may occur for various reasons. Failure typically happens during native code execution; for example, EAServer native code, JVM, or PBVM code. These issues are most difficult to trace when they occur on the client side. The following debugging suggestions may help you analyze the problem and enable you to provide enough information to the engineering team so it can either fix the issue or suggest other debugging steps:

- Look at all the logs carefully, including the JVM crash logs in the *bin* directory, to determine which operation could have caused the crash.
- Try to identify the steps that led up to the crash.
- Check the PATH and LIBRARY_PATH settings to verify that nothing unusual is present. You can do this by modifying the *run-server.bat*[*sh*] script to print the environment variables. LIBRARY_PATH, which is read at server start-up, should not include any JVM directories.
- Run EAServer in debug mode to see whether any helpful information is printed.
- If the previous suggestions do not help:

- Start EAServer using the -stopOnError flag. This flag suspends the JVM whenever a crash-causing failure occurs. A message displays in the console (or in a message box on Windows) indicating that a fault condition has occurred. Next, attach a debugger or other tool (such as pstack on Solaris) to get the stack trace of the entire process of the thread on which the crash occurs. See "Using stack traces" on page 8.
- Enable IIOP tracing to find which operations are causing the crash. This might be tedious to decipher in a multiuser environment. See "EAServer tracing" on page 9.
- Check memory usage to verify that the crash is not related to lack of memory on the native end.
- Check the Internet to see if other users have reported similar problems with the current version of the JVM.
- If a PowerBuilder component causes EAServer to crash, the pointers provided for crash analysis may be helpful.
- Try other (newer or older) versions of the JVM.

  **Note** This can be risky, because an untested JVM may cause some functionality to work incorrectly.

## More tips for debugging server failures

Server failure can take several forms:

- Server crashes on start-up
- Server fails at a specific point that is reproducible
- Server crashes intermittently

### Server crashes on start-up

To resolve or analyze a start-up crash:

1 Make sure that the PATH and CLASSPATH variables are correctly set.

  Verify which DLL or library versions have been loaded, and class load locations. For Windows, see "ListDLLs" on page 69. For UNIX, use pmap or another command or tool appropriate for your platform.

2 If you are operating in a cluster environment:

  - Check cluster start-up/check settings.

- Ensure all members have the same listener names and types.

3 Check the server log file. Besides reporting errors, the log file can also help determine how far along the start-up process was. The start-up tasks include:

   a Loading licensing files, check options

   b Loading network libraries for listeners

   c Initializing the JVM

   d Initializing various data structures

   e Loading server ClassLoader

   f Loading JCM (Java Connection Management) caches, connectors, and resources

   g Starting services:

      - Repository, GC, JCM, and Naming ServletService
      - User services; for example, the message service

   If there are no problems, the server begins accepting regular client connections, as indicated by the console message "Accepting connections."

4 In the server start batch file, set ECHO to on. This provides additional output and may help determine the point of failure.

5 Gather crash information, such as the load address. For Windows, see "ListDLLs" on page 69; for UNIX, use pmap or another command or tool appropriate for your platform.

   When the server crashes, identify the crash address using the console or log file. Use the address to identify the crash area or the module being executed. When an executable is launched, it tends to use the same set of base addresses, as virtual addresses normally do not change. This helps determine the specific product or application to investigate. The address may show that the crash occurred outside of EAServer; for example, in a system module. If the crash occurred in EAServer or a Sybase module, the address may be helpful to Technical Support, and may point to a known issue.

6 If EAServer crashes while running as a service, start EAServer in a console instead. A service typically runs under different resource constraints and permissions than a console.

7 If the message service is configured, set cms.debug to true.

### Server fails at a specific point that is reproducible

If the server fails at a specific, reproducible point:

1 Verify that the server started correctly, as indicated by these console messages:

```
Accepting Connections: iiop://host:2000
Accepting Connections: iiops://host:2001
Accepting Connections: iiops://host:2002
The Management Console can be accessed at http://host:8000/console
The Management Console can be accessed at https://host:8001/console
Accepting Connections: http://host:8000
Accepting Connections: https://host:8001
Accepting Connections: https://host:8002
Server Started
```

where *host* is the EAServer host machine.

2 Check the server log for errors, to assess when the crash occurred, and to determine areas for further investigation.

3 In the Management Console, check the MessageProfiler tab to find out which components are active, the connections being used, and so on—see “Runtime monitoring tools” on page 67.

4 Using the information from steps 1 and 2, turn on debug and trace flags for specific components, connections, and so on.

5 Explore what is happening in the application or component. Look for common application trouble spots—see “Generic issues” on page 24.

6 Check the component and server property settings.

7 Debug the component.

8 Check the server configuration, specifically:

- Patch levels—see “EAServer log” on page 4 for details.
- Resource and memory usage.
- DLL versions, library versions, and Java class loading—see “Verifying your configuration” on page 34 for library and class information, and “ListDLLs” on page 69.

9 Invoke a service component in the code just before where the crash is taking place to write information to the EAServer log—see “ListDLLs” on page 69:

- JVM properties
- Free and total memory
- EAServer monitoring data
- EAServer memory dump

10 Use the stack trace, dump file, and crash address for advanced troubleshooting—see “Stack traces, dump files, and core files” on page 74.

### Server crashes intermittently

If EAServer runs for a while, then crashes intermittently, use a similar plan as for reproducible crashes above.

- Ensure that sanity checks are enforced on the connection; otherwise, the cache may hand out invalid connections.
- Use a service component to periodically write information to the server log.
- Look for common application trouble spots.
- Monitor for memory leaks.
- Check peaks, maximums, and exhausted system resources.

## Server hangs

If EAServer runs for a while, then stops responding to requests, first follow the debugging suggestions described in “Server crashes” on page 16. If these do not solve the problem:

1 Check whether the issue exists only with specific types of requests (listeners, connections, and so on) or with everything (EAServer is not responding at all). If it is specific, continue with step 2; otherwise, go to step 4.

Check listeners, including IIOP, TDS, and HTTP as follows:

- Use the Management Console to connect to the IIOP listener port. A successful connection means that IIOP is working.

If you cannot connect, check to see whether the maximum number of IIOP or total sessions have been reached. If not, obtain a stack trace.

- Check the HTTP listener port at http://*machine_name*:8000 for default documentation. If the document is visible, HTTP is working. If you cannot view the default documentation:
  - Check whether the maximum HTTP or total sessions have been reached.
  - Turn on HTTP request logging.
  - If you are using a Web server redirector, set verbose logging in the configuration file.
  - Check the HTTP request log, HTTP error log, and the server log.

2 Check the data source:

- Use the Management Console to ping the data source. If ping fails, use isql (or a vendor-supplied tool for a non-Sybase database) to try a simple SQL request outside EAServer.

  Use a database monitoring tool to investigate what is happening on the database server side. Check locks held.
- In the Management Console, look at the server's DataSource tab. Verify that you have not reached the maximum number of connections for the data source. Check the number of connections: active, pooled, forced, peak, and so on.
- If you are trying to connect to Adaptive Server Enterprise, check that the log file is not full; this can cause EAServer to hang. Trace the connection using the Ribo utility to verify that the last SQL request passed. For information about using Ribo, see the jConnect for JDBC Installation Guide at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.jconnjdbc_6.05.jconnig/html/jconnig/title.htm.
- If you are making a JDBC connection, turn on the JCM debug flag to identify any SQL requests that may be causing the problem; try connecting through isql (or a vendor-supplied tool for a non-Sybase database) to check the response time.

3 Check whether a specific component or Web application may be responsible:

- Try another component.

- Review component properties. Is the component shared? Are variables set unexpectedly?
- Review the component code to look for:
  - A non-threadsafe sleep call
  - Synchronization for static variables
  - Proper release of resources
- Try using the Apache Tomcat servlet engine to host your Web components, delegating EJB requests to EAServer. See white paper #1016589: Using EAServer 4.0 with Apache Tomcat 3.3 at http://www.sybase.com/detail?id=1016589 for details.

4 If there is no response from any facility, including HTTP and IIOP listeners, connection caches, or components:

- Verify whether EAServer maximums are being reached; specifically, check if IIOP client sessions are at their maximum, or if the maximum thread limit (IIOP + HTTP) is reached.
- Check whether other tasks are running, and whether operating system peaks are being hit.
- Get a stack trace or dump. You must first kill the java process, run the core image, or generate the dump using a debugger. See Chapter 6, “Advanced Topics” for more information.

## Server disappears

If the server runs for a while, then disappears:

1 Check log files for errors. See “Logging and statistics” on page 4 for details.

2 On UNIX, check to see if a core file has been generated. On Windows, check for a *.log* or *.dbg* file.

3 Check the console for errors.

4 If there are no errors, restart the server to capture information next time.

- Start the server from the command line, and redirect the output:

  UNIX:

  ```
  start-server.sh > err.out 2>&1
  ```

Windows:

```
start-server.bat >info.out 2 >err.out
```

- On Windows, set the crash handler to *userdmp.exe* instead of Dr. Watson (*drwatson.exe* or *drwtsn32.exe*). See Chapter 6, "Advanced Topics," for more on debugging tools.
- Run the server in debug mode:

  start-server.sh[bat] [*server*] -debug

5 Implement a service component to periodically write information to the server log. See Chapter 4, "Creating Service Components," in the *EAServer Automated Configuration Guide*.

## Server slows or runs out of memory

If the server is slow to start, the problem may be the license manager, which may take one or two minutes to start.

If the server starts as expected, then slows down or runs out of memory, investigate these areas:

1 In the server log, search for "MEMORY MONITOR STATISTICS" to find the current Java memory usage and virtual memory usage.

2 In the Management Console, select Servers | <*Server Name*> | Statistics, and monitor the system to check pooling, peaks, maximums, sessions, and so on.

3 Check system sizing (virtual memory and paging file).

4 Use system tools to see which resource is failing. Check virtual bytes, CPU utilization, threads waiting, and so on. For each of these steps, see "Runtime monitoring tools" on page 67.

5 Review EAServer timeout properties. If no timeout is set, which is often the EAServer default, this leads to indefinite wait for transactions, components, or methods ().

6 Look for application trouble spots—see "Application issues" on page 24.

7 Check for memory leaks. For Java profiling, use Borland OptimizeIt. On Windows, use the Performance Monitor.

8 Implement a service component to periodically write information to the server log.

# Connection problems

If you have trouble connecting to EAServer, check these items:

- If your server's listeners use "localhost" for the machine name, rather than the actual network host address, you can connect only from clients running on the same machine, and connecting clients must use "localhost." To connect from other machines, change the server listeners to use the actual network name rather than "localhost."
- Some clients may have trouble resolving DNS host names to IP addresses. In this case, change the server's listeners to use the host IP address rather than the DNS host name.
- If you have trouble connecting from Java clients, check for exceptions related to Java class loading. See "Deploying and running Java clients" in Chapter 13, "Developing CORBA/Java Clients," in the *EAServer CORBA Components Guide* for details.

If you are trying to connect from a Java applet, check your Web browser's Java console for error messages.

# Application issues

This section describes common application issues for all component types.

## Generic issues

Common application problems for all component types include:

**Non-threadsafe sleep calls** Cause the server, instead of a specific thread, to sleep. Always use jagsleep.

**User-spawned threads** Use a thread monitor to allow EAServer to manage threads that must be spawned by a component. See "Monitoring threads" in Chapter 3, "Creating and Configuring Servers," in the *EAServer System Administration Guide*.

**ResultSets** You must explicitly free the structures that you use to process result sets. A ResultSet object is closed only when the Statement object that generated it is closed.

**Shared components** A single shared component instance services all client requests. A shared component can store data in instance variables. However, if the component's Concurrency option is also selected, you must add code to synchronize access to instance variables.

If a PowerBuilder component is Shared, disable Concurrency. PowerBuilder is thread-safe only at the session level.

**Lack of error handling** Code that does not check for errors, and invalid object references and calls are common problems. For more information, see Chapter 4, "Exception Handling."

**No timeout on stateful components, or no remove() to disconnect the client proxy** Before destroying the client proxy, deactivate the component instance using the SetComplete method. Do not leave a component instance bound to the client without a reference to it.

**Variables holding references** Set variables that hold references to null, particularly pooled components or connections.

## Java component issues

**Deploying from an EAR, JAR, or WAR** If you have trouble deploying from an EAR, JAR, or WAR, use the verifier tool to verify that it is valid. For details, see Verifier Tool at http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Tools11.html.

If you are having trouble deploying a Web application, check the J2EE requirements. If an application does not adhere to the J2EE rules, you may not be able to deploy it. For example, because they require local access, entity beans that participate in a container-managed relationship must all reside in the same EJB-JAR.

On Solaris and Linux, deploying a JAR file may fail if the class path includes "$JAGUAR," an environment variable that was used in versions of EAServer earlier than 6.0, and which therefore may include some 5.*x* classes in the class path.

**Static variables in Java** Static variables must be final or primitive, and cannot be larger than 32 bits. They can lead to potential race conditions: a variable is defined at the class level rather than the instance level; two threads try to modify a variable simultaneously. Race conditions are difficult to troubleshoot, and cause intermittent problems. You can use synchronization to prevent race conditions, but synchronization can cause performance bottlenecks.

**Synchronizing component skeletons** A skeleton class interprets component invocation requests and calls the corresponding method in your component with the parameter values supplied by the client. When a client sends an invocation request, the skeleton reads the parameter data and calls the Java method. When the method returns, the skeleton sends output parameter values, return values, and exception status to the client.

Problems can arise if component skeletons get out of sync. Generate a new skeleton class if you do any of the following:

- Install the component in a different EAServer package
- Change the name of the implementation class or move it to a different Java package
- Add a method to the component interface
- Delete a method from the component interface
- Change the signature of an existing method in the component interface

## C++ component issues

Incorrectly coded C++ components can corrupt EAServer memory. If you are having trouble with C++ components, Sybase suggests that you try moving them to another server and repository, so that if they crash, other components are not impacted.

## PowerBuilder component issues

If a PowerBuilder component causes EAServer to crash, the pointers provided for crash analysis may be helpful. If EAServer does not crash, insert debug statements into the PowerBuilder component.

**Migrating PowerBuilder components** When you migrate PowerBuilder components from EAServer 5.*x* to 6.0, the migration tool does not generate CORBA/Java stubs automatically. If stubs are required, you must generate and compile them manually:

1 From the command line, run:

```
idl-compiler.bat p_test.idl -f DJC_HOME\genfiles\java\src -java
```

where *DJC_HOME* is the EAServer installation directory.

2 Change to *%DJC_HOME%\genfiles\java\src\p_test*, and run:

```
JDK_HOME\bin\javac -d DJC_HOME\genfiles\java\classes -classpath
DJC_HOME\lib\eas-servr-14.jar;DJC_HOME\java\classes *.java
```

where *JDK_HOME* is the JDK installation directory

**Web DataWindow™ stability** You may see stability issues or hanging behavior in the Web DataWindow component, in some pre-9 PowerBuilder versions. See white paper #1023707: Web DataWindow Stability Issue at http://www.sybase.com/detail?id=1023707 for details.

**EAServer/PowerBuilder memory tuning** See white paper #1027319: EAServer/PowerBuilder Memory Tuning and Troubleshooting at http://www.sybase.com/detail?id=1027319.

**PBOnFatalError variable** The PBOnFatalError system environment variable allows you to specify whether EAServer should continue, restart, or shut down when an internal exception occurs in the PBVM. For more information, see “Unhandled PowerBuilder exceptions” on page 49.

**PBRollbackOnRTError variable** If a runtime exception is raised by a PowerBuilder component running in EAServer, the value of PBRollbackOnRTError determines the outcome of the transaction. If set to true, the transaction is rolled back; if set to false, the transaction is committed. After the transaction is either rolled back or committed, the exception is thrown back to the client.

The default behavior in PowerBuilder 8 prior to Build 10656 and in versions of PowerBuilder 9 prior to build 7151 is to commit the transaction.

**Threading models on Sun Solaris** Using a many-to-many threading model on Solaris may cause EAServer to hang or crash, if the server is highly stressed. Consider using a one-to-one threading model. See white paper #1026268: EAServer on Solaris - Troubleshooting Tip for Crashes or Hangs at http://www.sybase.com/detail?id=1026268.

**Performance guidelines** If you have trouble running PowerBuilder components under a heavy load, see "PowerBuilder component performance" in Chapter 3, "Component Tuning," in the *EAServer Performance and Tuning Guide*.

**Code sets** Use the following guidelines when deploying PowerBuilder clients or components to EAServer, and when troubleshooting issues related to code sets:

- In PowerBuilder clients that use char values greater than 127, specify the code set using the -ORBCodeSet property. The default code set (UTF-8) does not work, because PowerBuilder strings cannot handle 3-byte encodings. To specify the code set, set the Connection object's Options property; for example, to handle Korean characters in the eucksc code set, use the following syntax, where *myConnection* represents the Connection object:

  ```
  myConnection.Options ="ORBCodeSet='eucksc'"
  ```

- Specify the component- or server-level default code set.
- For PowerBuilder clients and components, verify that the specified code set is compatible with the operating system locale where they are running. That is, a client's code set must be compatible with the client's locale, and a component's code set must be compatible with the component's locale. The client's and component's locales need not be the same.
- If you encounter a character-conversion problem, verify that the character is valid in the selected code set, and that it has a well-defined encoding in Unicode; for example, the euro character is not valid in ISO 8859-1.

For more information about working with code sets, see white paper #1028793: Guidelines for Code Set Interoperability with PowerBuilder and EAServer at http://www.sybase.com/detail?id=1028793.

**Trace flags** All EAServer trace flags are described in "EAServer tracing" on page 9.

**Error handling** See "PowerBuilder error handling" on page 49.

## Avoiding memory leaks

Here are some ways to avoid memory leaks when designing your EAServer applications. Some points are more relevant to PowerBuilder applications and others to Java.

- Stateful objects have a one-to-one relationship with the client, so if the client disconnects, the object persists until the server times out. Stateless objects are preferable.
- Close database connections when finished; otherwise, the transaction manager cannot give connection resources to another requestor until a timeout occurs.
- Keep variable scope as small as possible. Limit the use of instance variables. Even if a component instance is not currently active, its storage persists for the lifetime of the instance.
- Use the appropriate event to create and destroy objects. If stateful, create instance objects in the constructor event and destroy them in the destructor event. If stateless, reset instance objects in the activate and deactivate events. In the deactivate event, set instance variables in Java components to null. When using DataStores in PowerBuilder components, reset the DataWindowObject to an empty string: `ds.dataobject = ""`, where *ds* is a data store reference to an instance variable.
- Minimize refreshing EAServer components and Web applications in production environments. Refreshing leaves the prior implementation loaded in memory, and excessive refreshing can overuse memory. When you do refresh components or Web applications in a production server, perform the refresh operation at the lowest level possible. For example, if you change one component, refresh that component only.

  **Note** Never refresh a server in a production environment, as this reloads all components and Web applications.

- Destroy any objects you create; do not rely on the garbage collector.
- Clean up after database activity, closing database connections and managing transactions to ensure correct life-cycle execution of components.
- Set blobs to null when finished.

## Applications that use Xerces

If a self-contained application that uses Xerces and other Axis classes deploys correctly but displays errors when running, the problem is a class-version conflict between the classes shipped with EAServer and those used by the application. To solve this problem, you can either:

- Use the disableResolveFirstBySystem option when you deploy an application, to tell EAServer to resolve classes using the internal class loader first, or
- Remove the application's copy of the classes. This may not work, because the application may run only with its own classes.

## Other design issues

For a discussion of the performance aspects of client applications, see Section 5, Client Applications, in white paper #1019504: EAServer Performance Tuning Techniques at http://www.sybase.com/detail?id=1019504.

# Security keys and certificates

You can configure EAServer to accept client connections over the secure protocols IIOPS and HTTPS by managing certificates and the keys in a keystore. See Chapter 11, "Managing Keys and Certificates," in the *Security Administration and Programming Guide*.

***Table 2-1: Common key/certificate questions***

| Questions | Answer |
|---|---|
| Where can I find private key and certificate information? | In the Management Console, select the server, then select the Security tab. The key and certificate information displays. The keystore holds all server-side certificates (private keys); the truststore holds the trusted certificates. |
| What types of keystores and truststores does EAServer support? | If the JDK with which EAServer is running is unmodified, the supported types are "pkcs12" and "jks." If a third-party plug-in is installed, additional types may be supported. |
| What tool can I use to maintain keystores and truststores? | EAServer supports the management tool keytool, which is a component of the JDK. See the keytool documentation at http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html. The keystore and truststore targets are defined on the server's Security tab. |

| Questions | Answer |
|---|---|
| How can I assign a certificate to a listener? | In the Management Console:<br>1 Select the listener you want to use.<br>2 On the General tab, note the name of the Security Profile.<br>3 In the left pane, expand the Security \| Profiles node, and select the security profile that the listener uses.<br>4 On the General tab, set the Certificate Label to the ID of the certificate you want the listener to use.<br>**Note** The certificate ID is a valid private key alias name that is available in the keystore. |

# Web server redirector plug-in issues

This section describes issues you may encounter if you use one of the Web server redirector plug-ins that are included with EAServer.

To access the *EAServer Installation Guide*, which describes how to install and configure the Web server redirector plug-ins:

1 Go to the Product Manuals site at http://infocenter.sybase.com.

2 Select EAServer 6.0, then select the *Installation Guide* for your platform.

## Apache and Sun Java System Web server redirectors

If you are using the Apache or Sun Java System redirector plug-in, you can trace requests by setting this directive in the redirector configuration file:

```
Connector.SessionId <ConnectorSessionId>
```

When this directive is set, the value of *ConnectorSessionId* is appended to the URL that is forwarded to EAServer. EAServer writes the URL to the server's HTTP request log, which can be helpful for debugging. For example, if you add this to the redirector configuration file:

```
Connector.SessionId ConnSID
```

EAServer writes this information to the HTTP request log:

```
10.22.85.66 - - "GET /TestHTTPS/?ConnSID=2696_000000000000 HTTP/1.0" 200 51
```

```
10.22.85.66 - - "GET /TestHTTPS/?ConnSID=2888_000000000000 HTTP/1.0" 304 0
10.22.85.66 - - "GET /TestHTTPS/?ConnSID=2889_000000000000 HTTP/1.0" 304 0
10.22.85.66 - - "GET /TestHTTPS/?ConnSID=2888_000000000001 HTTP/1.0" 304 0
10.22.85.66 - - "GET /TestHTTPS/?ConnSID=2889_000000000001 HTTP/1.0" 304 0
```

In this example, the Apache Web server process 2696 sent one request, process 2888 sent two requests, and process 2889 sent two requests. The connector's session ID is computed as:

process identifier of the Web server's process + request count

## Microsoft IIS Web server redirector plug-in

If you are having trouble using the IIS Web server redirector plug-in, verify that your environment is configured correctly.

❖ **Verifying the environment for an IIS redirector**

1. Verify these files exist in the EAServer *lib* directory:
   - *lijctssecct.dll*
   - *libjcc.dll*
   - *libjsybscl.dll*
   - *libjintl.dll*
   - *libjeas_iis.dll*
2. Verify that the PATH system variable includes the EAServer *lib* directory.
3. Verify that the WSPLUGIN_CONFIG_FILE system variable points to the location of the *redirector.cfg* file. For example, if *redirector.cfg* is located in *c:\windows\system32\inetpub*, the value of WSPLUGIN_CONFIG_FILE should be *c:\windows\system32\inetpub\redirector.cfg*.
4. Check the following logging settings in *redirector.cfg*:
   - connector.IIS.LogLevel should be set to "verbose" only when debugging, not when using the runtime DLLs.
   - connector.IIS.LogFile must be set to a valid drive and folder; otherwise, the plug-in may not load.
5. Look for error information in the log file that is defined by connector.IIS.LogFile (in *redirector.cfg*).

6 Expand the context path so the redirector handles all requests sent to the Web server. In *redirector.cfg*, set Connector.IIS.URLS to "/*".

7 Verify that entries in *redirector.cfg* have a corresponding protocol listener defined in EAServer. For example, the following entry in *redirector.cfg* requires that an EAServer HTTP protocol listener is defined with the host name "prodhost" and the port number 8000:

```
Connector.WebApp /* = http://prodhost:8000
```

8 Examine the log file to ensure that the host name is correct and the port number is valid. For example, given these entries in *redirector.cfg*:

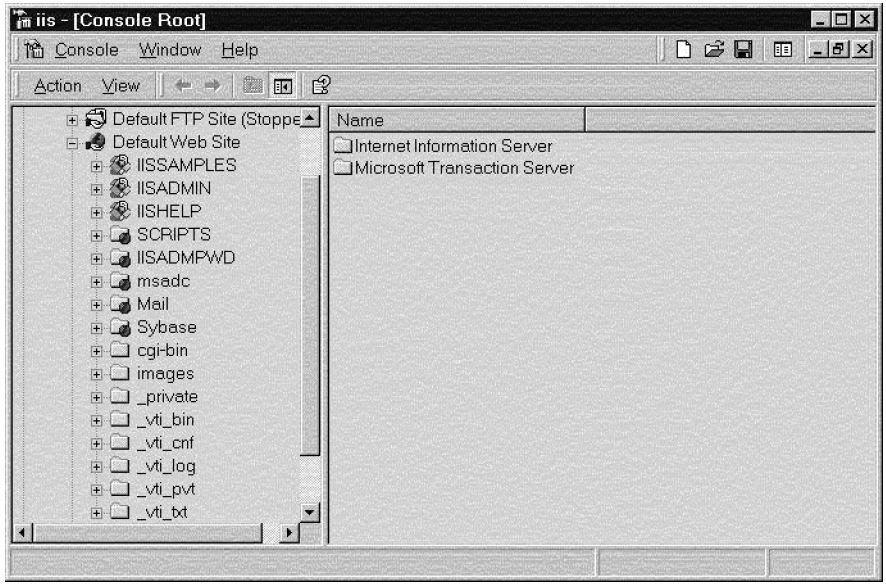
```
Connector.WebApp /enowcorporate = http://prodhost:8000

Connector.IIS.LogLevel inform
```

the log file should display:

```
Wed May 21 16:53:58 2003 INFO: ws: 1de1090, URL: [http://prodhost:8000],
protocol: [http], host: [prodhost], port: [8000] down_time:0
```

9 Using the IIS administration tool, verify that the icon for the virtual directory you are using looks like the Sybase folder in Figure 2-1. If the icon looks as expected, skip to step 11.

***Figure 2-1: IIS administration tool***



10 If the icon for your virtual directory looks like the IISADMIN icon, instead of like that of the Sybase folder:

   a Open the Properties dialog box for your virtual directory.

   b Select the Virtual Directory tab, and to the right of Application Settings Name, click Remove.

   c Restart IIS.

11 If none of the previous steps help identify the problem, try running the debug version of the plug-in, and check the log for error information.

# Configuration issues

If you are having trouble starting EAServer, connecting from a client to the server, or invoking components from the server, first verify your configuration and try to identify any runtime errors following the instructions below. If all else fails:

- Shut down and restart the server.
- Perform a full rebuild of applications.

## Verifying your configuration

For all platforms, verify:

- You can ping the data source.
- The deployment properties of the application components are correct. See Chapter 9, “Importing Application Components,” in the *EAServer System Administration Guide*.
- Java classes are loaded from the correct locations. See Chapter 10, “Configuring Java Class Loaders,” in the *EAServer System Administration Guide*.

  You can find all occurrences of a class in BOOTCLASSPATH and CLASSPATH using the ClassSearch utility, which is described in white paper #1017251: ClassSearch Utility at http://www.sybase.com/detail?id=1017251.

- Maximum values and limits are set appropriately for the production environment:
  - EAServer properties (data sources, sessions, threads, and so on)
  - JVM
- The listener ports are not already in use and that no two EAServer listeners are defined to use the same port number.
- The versions of the PBVM on the server and the PowerBuilder client match. Check all files, especially *libjcc.dll*, which may be shipped with both PowerBuilder and EAServer releases, ESDs, and EBFs.

**Multiple versions of libjcc.dll** EAServer includes *libjcc.dll* in the EAServer *lib* directory. PowerBuilder may include the file as well, in *%SYBASE_SHARED%/PowerBuilder*, since this file is needed on PowerBuilder client installations. Sybase recommends that you not copy the *libjcc.dll* that is provided with PowerBuilder to the server as part of a PBVM setup. Even if you have the latest PowerBuilder patch, use the EAServer *libjcc.dll.*

Problems can result if either the EAServer application server or the Management Console uses a different version than its own (such as the one provided with PowerBuilder). This used to be a problem when EAServer looked at the system path variable. However, now that EAServer sets up the path in the *start-server* script, the problem occurs only when users modify either the script or environment.

Solaris platforms

Verify that LD_LIBRARY_PATH includes:

```
$DJC_HOME/lib:$ASANY9/lib:$DJC_HOME/bin:
$DJC_HOME/intersolv/lib:$DJC_HOME/lib/debug:
$LD_LIBRARY_PATH
```

Other UNIX platforms

Add the location of the X-Window xterm utility to your PATH variable. For example:

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

If you cannot start the server, verify that these environment variables are properly set:

- PATH – must include *$DJC_HOME/bin*.
- DJC_HOME – specifies the location of the EAServer installation directory.

- ASANY9 – specifies the location of the Adaptive Server Anywhere directory.

Windows platforms

If the PATH or CLASSPATH settings do not include the required directories, or if the settings are too long, you may experience problems.

Look at *%DJC_HOME%\bin\djc-setenv.bat* and *%DJC_HOME%\bin\run-server.bat* to investigate problems with either the PATH or the CLASSPATH variable.

## Running EAServer as a service

If you experience problems running EAServer as a service, try running it from the console. The service installed by default runs under the LocalSystem account, and access to printers and other resources is limited. These resource allocation problems may not occur when running the server from the console.

You can also configure the service to run under the account of the user who installed the service, which may fix allocation problems.

❖ **Configuring the service account**

On Windows 2000 and Windows XP:

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

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

3 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.

4 Click Apply.

# System-level issues

This section describes system-level issues that may affect EAServer.

## Operating system issues

Operating system issues, sometimes obscure and apparently unrelated, may adversely impact EAServer. Recent examples include:

- Solaris timers, which may affect looping service components
- AIX signals, which may lock up threads

## CPU sizing

Check that your machine is sized properly to handle peak loads.

To determine a reasonable upper value of simultaneous client connections before the server starts "thrashing," see white paper #1019577: CPU Sizing for Concurrent Client Connections to EAServer at http://www.sybase.com/detail?id=1019577.

## Multiprocessors

Numerous EAServer Change Requests (CRs) have already addressed multiprocessor issues. Check that you have the latest relevant patches.

Not all operating systems fully utilize multiple CPUs. Depending on your operating system, you may need to bind EAServer to each CPU to take advantage of multiple processors. See white paper #1010600: Binding a PowerBuilder Process to a CPU on Windows NT or Sun Solaris at http://www.sybase.com/detail?id=1010600 for details. This document is written for PowerBuilder but has wider application.

## UNIX file descriptors

On UNIX platforms, concurrent client connections to EAServer are limited by the operating system limit for the number of file descriptors that one process can open. Before you start the server, use the ulimit command to set the file descriptor limit in the shell where you will start the server.

See your UNIX system documentation for details about ulimit.

## Windows virtual bytes

On Windows, every process has a default limit of 2G virtual bytes. The default limit may not be sufficient in some conditions, such as high-load scenarios with many instances of PowerBuilder components. The EAServer process may reach on out-of-memory condition when the virtual byte limit is reached.

Some Windows server editions, such as the Advanced edition, allow you to configure a higher limit such as 3GB.

To determine whether the EAServer process is reaching the virtual bytes limit, see “Evaluating Windows memory” on page 67.

To increase the virtual bytes limit for the EAServer process, use the Windows *imagecfg.exe* utility. For more information about the utility, see the Microsoft Knowledge Base article #171793: Information on Application Use of 4GT RAM Tuning at http://support.microsoft.com/kb/171793/.

## Running on a 64-bit platform

To run EAServer 6.*x* on a 64-bit platform:

1 Download the 64-bit version of the JDK.

2 Start EAServer using the -arch64 flag.

If you will always run EAServer in 64-bit mode, set up these flags in *local-sevenv.bat* (Windows) or *local-sevenv.sh* (UNIX):

- DJC_RT_DEFAULT=15
- DJC_JDK_DEFAULT=15
- DJC_ARCH_64=true

To use JDK 1.6, replace “15” with “16”.

CHAPTER 3

# Performance Issues



## Resources

A number of documents are available to help you improve performance for EAServer applications. Access the EAServer 6.0 document collection at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.eas_6.0/title.htm. The collection includes:

- *EAServer Performance and Tuning Guide*

  This document covers all key aspects of performance, including:

  - General server tuning
  - Clusters
  - All components, including service, Java, EJB, JSP, Servlet, C/C++, and PowerBuilder components
  - Message service
  - Client applications
  - Database access
  - Web Services Toolkit
  - Runtime monitoring
- *What's New in EAServer*

  Highlights new features in version 6.0.
- *EAServer System Administration Guide*

Chapter 8, “Load Balancing, Failover, and Component Availability,” explains how to optimize performance for an EAServer cluster by adjusting the load across the servers.

Chapter 11, “Runtime Monitoring,” describes how to use the Management Console and the server log files to monitor EAServer performance.

- *EAServer EJB User’s Guide*

  Chapter 2, “Deploying and Configuring EJB Components,” explains how to increase performance for EJB entity beans with object and query caching.

CHAPTER 4

# Exception Handling



## Overview

As a general rule, developers should always include exception handling in their code to catch and output error message details to help with troubleshooting. For example:

```
try {
   // some logic here
} catch (Exception e) {
System.out.println("Exception caught in mycomponent/mymethod: "
                   + e.getExplanation() + e.toString());

   // depending on exception type, you may want a stack trace
   e.printStackTrace();
   // return or retry as appropriate
}
```

The sections below provide information about handling exceptions for different EAServer component modes.

**Note** A listing of commonly encountered server errors appears in Chapter 5, “Common Error Messages.”

# Error handling in CORBA Java components

Handle errors occurring during component execution gracefully, as follows:

1 Write detailed descriptions of the error to the log. This will help you debug the problem later. You can call any of the System.out.print methods to write to the log (the output is redirected).

2 If the error prevents the current transaction from completing, roll it back; for details, see "Set transactional state" in Chapter 12, "Developing CORBA/Java Components," in the *EAServer CORBA Components Guide.*

3 Throw an exception with a brief, descriptive message appropriate for display to an end user of the client application.

Java components can record errors or status messages in the server log file. Writing to the log creates a permanent record of the error, and log messages can be automatically stamped with the date and time that the message was written. Call any of the System.out.print methods to write to the log.

You can also throw an uncaught exception. Ideally, any exception thrown by your component should be a standard CORBA IDL exception or a user-defined IDL exception; the latter must be listed in the raises clause of the IDL method definition and the throws clause of the equivalent Java method declaration. All exceptions are forwarded to the client, but only exceptions defined in IDL can be rethrown by the client stub as a duplicate of the server-side exception.

CORBA ORB and EAServer EJB clients receive forwarded exceptions differently:

- CORBA ORB clients rethrow any exception defined in IDL as a duplicate of the original exception. Other exceptions are rethrown as the standard CORBA exception UNKNOWN.
- EAServer EJB clients rethrow any server exception as a JException instance with the message text returned by calling toString on the original exception.

# Handling exceptions in CORBA Java clients

The client-side ORB throws two kinds of exceptions:

- CORBA system exceptions – defined in the CORBA specification.

- User-defined exceptions – defined in the component's IDL definition.

## CORBA system exceptions

The CORBA specification defines the list of standard system exceptions. In Java, all CORBA system exceptions extend org.omg.CORBA.SystemException. System exceptions are unchecked exceptions (they extend java.lang.RuntimeException). The Java compiler does not require that you catch CORBA system exceptions. However, some exceptions can occur in a well-behaved program. For example, the Session.loookup call throws a NO_PERMISSION exception when you request a component instance and the user lacks permission to instantiate that component. You may want to trap the exceptions shown in the code fragment below:

```
try
{
    // invoke method(s)
    ...
}
catch (org.omg.CORBA.COMM_FAILURE cf)
{
    // If this occurs when instantiating a Manager
    // instance, the server is likely down or has run
    // out of connections. You can retry the connection
    // if desired.
    //
    // If this occurs after a method call, you
    // can retry the call (or the transaction call
    // sequence for a stateful component).
    ...
}
catch (org.omg.CORBA.TRANSACTION_ROLLEDBACK tr)
{
    // A component on the server aborted the EAServer
    // transaction, or the transaction timed out.
    // Retry the method call(s) if desired.
    ...
}
catch (org.omg.CORBA.OBJECT_NOT_EXIST one)
{
    // Possibly try to create another instance. Check
    // that the package and component are installed
    // on the server.
    // Received when trying to instantiate a component
```

```
        // that does not exist. Also received when invoking
        // a method if the object reference has expired
        // (this can happen if the component is stateful
        // and is configured with a finite Instance Timeout
        // property). Create another instance if desired.
        ...
    }
    catch (org.omg.CORBA.NO_PERMISSSION np)
    {
        // Tell the user they are not authorized
        ...
    }}
    catch (org.omg.CORBA.SystemException se)
    {
        // Catch-all clause for any CORBA system exception
        // that was not explicitly caught above.
        // Report the error but don't bother retrying.
        ...
```

**Note** Not all of the possible system exceptions are shown in the example. See CORBA/IIOP 2.3 specification for a list of all the possible exceptions.

## User-defined exceptions

User-defined exceptions are defined in the component's IDL definition. For example, you might define OverdrawnException to be thrown by methods that withdraw money from a bank account. In Java, all user-defined exceptions extend org.omg.CORBA.UserException.

In Java, IDL user-defined exceptions are checked exceptions; if the IDL definition of a method contains a raises clause, the equivalent Java stub method will have a throws clause that lists the equivalent Java exceptions. For example, consider the IDL definition below:

```
module MyModule {
  exception MyException
  {
    string reason;
  };

  interface MyIntf {
    boolean throwException
```

```
        ( in boolean yes_no )
        raises (MyException);
    };
};
```

The equivalent Java throwException method is:

```
boolean throwException (boolean yes_no)
    throws MyModule.MyException;
```

# Error handling in CORBA C++ components

The client-side ORB throws two kinds of exceptions:

- CORBA system exceptions – defined in the CORBA specification.
- User-defined exceptions – defined in the component's IDL definition.

## CORBA system exceptions in C++

The CORBA specification defines the standard system exceptions. In C++, all CORBA system exceptions are mapped to a C++ class that is derived from the standard SystemException class defined in the CORBA module. You may want to trap the exceptions shown in this code fragment:

```
try
{
... // invoke methods
}
catch (CORBA::COMM_FAILURE& cf)
{
... // A component aborted the EAServer transaction,
   // or the transaction timed out. Retry the
   // transaction if desired.
}
catch (CORBA::TRANSACTION_ROLLEDBACK& tr)
{
... // possibly retry the transaction
}
catch (CORBA::OBJECT_NOT_EXIST& one)
{
... // Received when trying to instantiate
   // a component that does not exist. Also
```

```
    // received when invoking a method if the
    // object reference has expired
    // (this can happen if the component
    // is stateful and is configured with

    // a finite Instance Timeout property).
    // Create a new proxy instance if desired.}
}
catch (CORBA::NO_PERMISSSION& np)
{
... // tell the user they are not authorized
}
catch (CORBA::SystemException& se)
{
... // report the error but don't bother retrying
}
```

Not all of the possible system exceptions are shown in this example. See the CORBA/IIOP 2.3 specification for a list of all the possible exceptions.

## User-defined exceptions in C++

In C++, all CORBA user-defined exceptions are mapped to a C++ class that is derived from the standard UserException class defined in the CORBA module. For more information, see "User-defined IDL datatypes" and "User-defined exceptions" in Chapter 3, "Using CORBA IDL," in the *EAServer CORBA Components Guide*.

User-defined types must exist in the EAServer IDL repository before you can use them in interface declarations.

# ActiveX clients

EAServer 6.0 does not support ActiveX components. ActiveX CORBA clients running with EAServer 5.*x* can access EAServer 6.0. Use EAServer 5.*x* to generate *.tlb* and *.reg* files to use with your ActiveX-enabled IDE.

# Using error pages

Error pages allow you to customize the response that the server sends to Web clients when an error occurs.

## Error pages for Web applications

When the servlet engine detects an error or catches an exception thrown by a servlet, it searches for a corresponding error page to handle the response. You can specify HTML files to send in response to HTTP error codes and to Java exceptions thrown in JSPs or servlets.

This example illustrates how to declare an error page for a Web application in the deployment descriptor:

```
<error-page>
  <error-code>404</error-code>
  <location>/etc/404.html</location>
</error-page>
```

The location is the path relative to the Web application's context root. For example, */etc/404.html* corresponds to this file in your EAServer installation directory, where *web-app* is the name of the Web application:

```
deploy/webapps/web-app/etc/404.html
```

## Error pages for JavaServer Pages

When a client request is processed, runtime errors can occur in the body of the implementation class for a JSP or in Java code that is called by the page. You can handle these exceptions in the JSP code using the Java language's exception mechanism.

Uncaught exceptions

Use an error page that you specify using a page directive to handle exceptions that are thrown from the body of the implementation class and are not caught. Both the client request and the uncaught exception are forwarded to the error page.

To specify an error page for a JSP, set its errorPage attribute to the URL of the error page in a page directive:

```
<%@ page errorPage="ErrorPage.jsp" %>
```

The java.lang.Throwable exception is stored in the javax.ServletRequest instance for the client request using the putAttribute method, using the name javax.servlet.jsp.jspException.

## Using an error page JSP

If you specify a JSP as the error page, you can use its implicit exception variable to obtain information about the exception. The exception variable is of type java.lang.Throwable and is initialized to the throwable reference when the uncaught exception is thrown.

To specify an error page for a JSP, set its errorPage attribute to the URL of the error page in a page directive:

```
<%@ page errorPage="ErrorPage.jsp" %>
```

To define a JSP as an error page, set its isErrorPage attribute to true in a page directive:

```
<%@ page isErrorPage="true" %>
```

This sample error page JSP uses the exception variable's toString method to return the name of the actual class of this object and the result of the getMessage method for the object. If no message string was provided, toString returns only the name of the class.

The example also uses the getParameterNames and getAttributeNames methods of the request object to obtain information about the request.

```
<%@ page language="java" import="java.util.*"
    isErrorPage="true" %>
<H1 align="Center">Exceptions</H1>
<br><%= exception.toString() %>
<%! Enumeration parmNames; %>
<%! Enumeration attrNames; %>
<br>Parameters:
<%  parmNames = request.getParameterNames();
    while (parmNames.hasMoreElements()) {
%>
    <br><%= parmNames.nextElement().toString() %>
<%
    }
%>
<br>Attributes:
<%  attrNames = request.getAttributeNames();
    while (attrNames.hasMoreElements()) {
```

```
%>
        <br><%= attrNames.nextElement().toString() %>
<%
    }
%>
```

# PowerBuilder error handling

The PowerBuilder documentation set includes extensive details on handling exceptions. Here are some useful references:

- "Exception Handling in PowerBuilder" in Chapter 3, "Selected PowerScript Topics," in the *Application Techniques* manual
- "Testing and debugging the component" in Chapter 24, "Building an EAServer Component," in the *Application Techniques* manual
- "Handling errors" in Chapter 25, "Building an EAServer Client," in the *Application Techniques* manual
- "Troubleshooting connections" in Chapter 25, "Building an EAServer Client," in the *Application Techniques* manual
- "Handling DataWindow Errors" in Chapter 2, "Using DataWindow Objects," in the *DataWindow Programmer's Guide*

To access the PowerBuilder documentation online, go to the SyBooks Web site at http://infocenter.sybase.com/help/index.jsp. From the PowerBuilder collection, select the book title.

## Unhandled PowerBuilder exceptions

An unhandled fatal exception raised by a PowerBuilder component running in EAServer, can cause the PBVM to become unstable, resulting in unpredictable behavior and unforeseen problems with the PBVM and EAServer. This scenario is unlikely, but possible. You may want to restart or shut down the server, rather than allow EAServer to continue running in an unstable state.

The PBOnFatalError system environment variable allows you to specify the action you want EAServer to take when an unhandled exception is raised in the PBVM. The PBOnFatalError variable is supported in PowerBuilder 8.0.4 (Build 10501) and PowerBuilder 9.0.1 (Build 6533) maintenance releases, and later. These are the values you can assign to the PBOnFatalError variable:

| Value | Resulting behavior when a fatal error occurs in a PowerBuilder component |
|---|---|
| continue | EAServer continues running, and a CORBA_TRANSACTION_ROLLEDBACK exception is thrown. This is the default value. |
| restart | EAServer restarts automatically. |
| shutdown | EAServer shuts down. |

CHAPTER 5

# Common Error Messages



## Introduction

This chapter lists the errors most commonly encountered during EAServer operation. It provides the context for each error and troubleshooting tips as applicable.

When an error is raised, try to determine where the error came from: is it an EAServer error, or was it passed to EAServer from another source such as a database or the Java virtual machine?

Some errors include a source indicator. For example:

| Source | Meaning |
|---|---|
| SRVLIB | Generated by EAServer |
| DEBUG | Generated by a component when the debug property is enabled |
| TRACE | Generated by a component when the trace property is enabled |
| CORBA | Generated by CORBA client ORB |

# Error messages

This section lists common messages and their explanation, with possible causes and tips on how to resolve them.

**Note** For simplicity, only the key portions of the error messages are provided. The actual message text may contain a prefix such as “Exception” or “System exception,” which is not included in these listings.

***Table 5-1: Server messages***

| Message text (or text fragment) | Explanation |
|---|---|
| Cannot find interfaces file | See “Connection problems” on page 24 and “Configuration issues” on page 34. |
| Cannot find localization files | See “Connection problems” on page 24 and “Configuration issues” on page 34. |
| Cannot start network listener | The main IIOP listener may be configured incorrectly. Check the console and log files for listener failures. Use EAServer Manager to verify the correct listener properties. Try to connect to the server with another application, using the same protocol.<br>See “Configuration issues” on page 34 for information about resolving start-up problems. |
| ClassCastException error | If you see NamingContext exceptions with root-cause exception ClassCastException when calling javax.naming.InitialContext.lookup, check for the following problems:<br>• You may be casting to an incorrect type (check the class name of the object returned by lookup).<br>• Your component has refresh enabled, and the custom class list does not contain some required classes.<br>• Your component has refresh enabled, and calls a component that has refresh disabled, or vice-versa. |
| com.sybase.jdbc.SybSQLException | When EAServer starts a transaction, it puts the current connection into chained mode. By default, Adaptive Server Enterprise runs procedures in unchained mode. Use sp_procxmode to change the mode of the stored procedure; for example:<br>`sp_procxmode "sp_myproc", "anymode" go`<br>You can run sp_myproc only in unchained transaction mode. The SET CHAINED OFF command causes the current session to use unchained mode. |

| Message text (or text fragment) | Explanation |
|---|---|
| CORBA marshall exception | You may have tried to return a NULL object, which CORBA forbids.<br>This error may also occur when you attempt to access an ActiveX component from Visual Basic if you have not explicitly initialized a structure field that uses complex types. Initialize fields of complex types such as struct, union, object, date, time, or timestamp. If you do not initialize these fields before passing the union as an EAServer method parameter or return value, the ActiveX dispatcher throws a marshalling exception. Fields of other types are implicitly set to a default value. |
| CORBA:: NO_PERMISSION | The user is not authorized to perform the requested operation. For example, the Session.loookup call throws a NO_PERMISSION exception when you request a component instance and the user lacks permission to instantiate that component.<br>Check permissions for the component, package, and user.<br>A CORBA::NO_PERMISSION exception can also mean that an attempt was made to access an object from a less secure session than it was originally created with. If the original proxy instance was created by connecting to a secure port with a client-side SSL certificate, the proxy must be deserialized in a session that connects using the same client certificate and equal or greater security constraints.<br>For example, if you create an object with a session that uses 128-bit SSL encryption, serialize the object, then later try to deserialize the object during a session that uses 40-bit SSL encryption, the ORB throws the CORBA::NO_PERMISSION exception. Access is allowed when objects created using a less secure session are later accessed using a more secure session. |
| CORBA::NO_RESOURCE_EXCEPTION | This exception is raised if a component request arrives when the maximum number of instances exist, all are busy, and the blocking time expires. (com.sybase.jaguar.component.objects specifies the maximum number of instances, and the Resources/Maximum Wait property specifies the blocking time.)<br>Network latency between client and server is not included in the measured method execution time. For C++ components running in an external process, the measured time includes interprocess communication latency. |

| Message text (or text fragment) | Explanation |
|---|---|
| CORBA::OBJECT_NOT_EXIST | This message can mean that the package or component is not installed on the server.<br>It may also be received when a method is invoked but the object reference has expired, which can happen if the component is stateful and is configured with a finite Instance Timeout property. When the timeout period is exceeded for a component instance, EAServer deactivates the component and invalidates the client's object reference. If the client attempts another method invocation, the client-side ORB throws the CORBA::OBJECT_NOT_EXIST exception. At this point, the client must create a new proxy instance for the component if it wants to continue. |
| CORBA::TRANSIENT | For EJB clients, this exception is the root cause of the java.rmi.RemoteException thrown by the EJB stub.<br>This message is associated with concurrency control; it may indicate a rollback due to an optimistic update failure. The default optimistic update check is to compare the old values with the new values. Someone else may have updated the data between your ejbLoad and ejbStore.<br>Use one of the following corrective actions:<br>• Set the property com.sybase.jaguar.component.debug=true to get more information; or<br>• Use a timestamp column in your database table; or<br>• Disable optimistic updates as appropriate for your application.<br>See Chapter 4, "EJB CMP Tuning," in the *EAServer Performance and Tuning Guide* for more information on concurrency control. |
| Could not start thread | The EAServer maximum threads property must include thread requirements of the entire server, including the message service thread pools. You can set this property on the HTTP Config tab in the Server Properties dialog box. |
| CtsComponents::CreateException | See javax.ejb.CreateException error. |
| CtsComponents::FinderException | See javax.ejb.FinderException error. |

| Message text (or text fragment) | Explanation |
|---|---|
| DBMS is not supported in your current installation | This is a PowerBuilder error message. The probable cause is that the value of SQLCA.DBMS is not set.<br><br>In PowerBuilder component code, you typically use a series of lines to set up database connection parameters using a PowerBuilder transaction object. For the default transaction object, SQLCA, you must set the value of SQLCA.DBMS to a string that begins with one of: ODB, JDB, SYJ, O90, O84, or O73. The definition must come before this statement:<br><br>`CONNECT USING SQLCA;`<br><br>PowerBuilder uses the DBMS string to build the DLL or library name that it needs.<br><br>If you are using a transaction object other than SQLCA, verify that its DBMS property is set correctly, before calling the connect statement. |
| Deployment error accessing server <*server_name*> at port 9000 | This is a PowerBuilder error message.<br><br>Connect to EAServer Manager to confirm that EAServer is running.<br><br>Confirm that EAServer is listening on the server name and port mentioned in the error. An IIOP listener must be configured on that port and server name.<br><br>Check the EAServer user ID and password specified on the Server tab in the Component Generator properties.<br><br>Check the path specified in the PowerBuilder Dynamic Library Name text field on the Libraries tab in the Component Generator Properties. |
| Deployment error functions using ANY type arguments or an ANY return type not supported. Correct the following for component <*component*>:<*method*>. | This message applies specifically to the ANY datatype, which is not supported for public instance variables, function arguments, or function return values. The remedies are the same as described for "Deployment error or warning (from PowerBuilder) SYSTEM variables not supported." |
| Deployment error or warning (from PowerBuilder): SYSTEM variables not supported | Public instance variables and arguments to public functions can be any of:<br><br>• Standard datatypes<br>• Structures<br>• Custom class user objects that have been deployed as EAServer components<br>• ResultSets<br><br>If you are using system datatypes (transaction, data store, and so on) as instance variables, declare them as protected or private. If you are using system datatypes as function arguments, declare the function as protected or private. |

| Message text (or text fragment) | Explanation |
|---|---|
| Incorrect password or tampered keystore | This problem occurs on Windows more often than on UNIX.<br>On Windows, the problem is either that the password for the user ID has been changed, or the user has logged on with a different ID.<br>If the problem occurs on UNIX:<br>1 Change to *Repository/Instance/com/Sybase/djc/server/ApplicationServer* in your EAServer installation.<br>2 Using a text editor, open the server properties file; for example, *<machine_name>.properties*.<br>3 Find the truststorepassword property. Typically, an asterisk follows the equal sign, which signifies that this is a protected property. Change the value of truststorepassword to the password for the truststore.<br>4 Find the keystorepassword property, and change the value to the password for the keystore.<br>5 Save and close the file, then restart the server.<br>6 In the Management Console, on the server properties page, select the Security tab, then update the keystore and truststore passwords to make the passwords protected properties again. |
| Instruction at <*location*> referenced memory at <*location*>, the memory could not be written | See “Server crashes, hangs, or disappears” on page 16. |
| INVALID_TRANSACTION | If you are using the message service, this exception occurs if you try to use two concurrent threads within a single transacted session to send or receive messages. You must create a separate transacted session for each thread that you use to send or receive messages. This restriction applies to messages received synchronously or asynchronously; that is, regardless of whether you call “receive” or use a message listener. |
| javax.ejb.CreateException OR CtsComponents::CreateException | When instantiating an entity bean proxy, call a finder method first if you are unsure whether an entity bean’s data is already in the database. Create methods throw a javax.ejb.CreateException exception if you attempt to insert a duplicate database row. |
| javax.ejb.FinderException OR CtsComponents::FinderException | EJB finder methods return instances that match an existing row in the underlying database based on the lookup parameters passed to the method.<br>Finder methods throw javax.ejb.FinderException if no rows match the specified search criteria. |

| Message text (or text fragment) | Explanation |
|---|---|
| java.lang.NoClassDefFoundError | A class definition could not be found. Make sure the class file is in the CLASSPATH.<br>If necessary, set the com.sybase.jaguar.server.classloader.debug property to true to enable class loader tracing while you troubleshoot class loading issues. Remember to reset the property to false when you are finished. |
| java.lang.UnsupportedClassVersionError | The JDK version that is used at runtime is different from the JDK version that was used to compile the classes. Typically, the JDK version used to compile the classes is earlier than the version used to run the classes.<br>Verify that the JDK version used by deploy or other commands is the same or later than the JDK version that was used to compile the classes. |
| javax.naming.NamingException | The lookup method throws javax.naming.NamingException if the bean JNDI name cannot be resolved or the home interface proxy cannot be created. Reasons include:<br>• The server address specified with the Context.PROVIDER_URL property is incorrect or the server is not running.<br>• Authentication with the specified credentials failed.<br>• The bean is incorrectly configured on the server. For example, a skeleton has not been generated, or the bean's properties specify the wrong implementation class.<br>• Check the EJB references in the Management Console to verify that the values are correct.<br>Check the server's log file if the cause of the error is not clear from the exception's detail message. |
| JCM Caught Throwable: java.sql.SQLException: Error: Current enlistment requires 2PC Resource and No 2PC Resource Configured for Cache:SybaseJMS | A component is attempting to write a JMS message to an EAServer message service topic or queue, without using the same connection cache. The component must use the same connection cache as the EAServer message service. |
| jmsException | An exception has occurred with the message service. The ExceptionListener provides access to the error information—see the *EAServer JMS User's Guide*.<br>You can also output debug information for the message service using the com.sybase.jms.debug property. |
| No such file libpb90x.so: not found | This is the same problem as described in "DBMS is not supported in your current installation." |

| Message text (or text fragment) | Explanation |
|---|---|
| OBJECT_NOT_EXIST | Most commonly, this is because the server component cannot be instantiated or an instance is no longer available. Verify that:<br>• The specified component is installed in the specified EAServer package.<br>• The specified package is installed in the server.<br>• The Java class, Windows DLL, or UNIX shared library that implements the component is available.<br>• If you are instantiating a Java component, the component's skeleton class is available.<br>This error may also occur when a component's cache size is not large enough, causing clients to experience cache overflow errors. When this happens, the least recently accessed instance is removed from the cache. If a client attempts to invoke an instance, the client receives a CORBA::OBJECT_NOT_EXIST exception.<br>The error may also be raised when more than one client simultaneously tries to receive a message from a message service queue that is not shared. |
| NullPointerException | For in/out parameters to CORBA Java component methods, you must pass a non-null value for the parameter input value. Otherwise, method calls fail and throw NullPointerExceptions. Use output parameters in the method definition if the parameter's input value does not matter. |
| ObjectKey::init: <servername>.cycle create failed: No such file or directory | This error may be seen on server start-up. Causes include:<br>• Your installation does not have a repository, or<br>• The repository is corrupt, or<br>• Repository/Server directory is not writeable.<br>**Note** *servername.cycle* is not required for the server to start. If the file does not exist, it is automatically created when you first start the server. |
| SetDwObject (*PBL_name*, *DataWindow_name*) failed = -1 | Verify the HTML DataWindow property in the DataWindow painter.<br>Verify that the PBL (PowerBuilder library) or PBD (PowerBuilder dynamic library) is in the system PATH of the server machine. |
| srv__read_packet: Protocol error occurred: length in header (21536) more than packet size(512) | Verify that the listener port matches the client protocol. |

| Message text (or text fragment) | Explanation |
|---|---|
| SRVLIB message: Net-Library routine net_listener (host:port) failed in srv_start_listeners<br><br>Network error: status = 23 - Net-Lib protocol driver call to register a listener failed | The listener is currently in use. Network listeners must be unique for the server machine. If the server runs as a Windows service, verify that you did not attempt to start another instance of the same server.<br><br>To fix the problem, modify the listener properties in the repository and restart the server. Be sure there is only one instance of a specific server running. |
| [SySAMLicenseManager] msgId: 131249, message: Failed to obtain 1 license(s) for EAS feature with properties 'PE=AE;LT=CP' | EAServer is configured to get a particular type of license, but the available license is not the correct type. To avoid this error, correct the license configuration in *default.properties*, located in */Repository/Instance/com/Sybase/djc/lm/FeatureManager*. Verify that the product edition and license type match the values in the license file. |
| SystemException: CORBA::COMM_FAILURE | Possible reasons for this error include:<br>• The server is down.<br>• The server has run out of connections.<br>• You did not specify the correct host name or listener address. |
| Topic is not available: Send Failure: …. org.omg.CORBA.COMM_FAILURE: java.io.Exception. | Message service topic is not available. Most likely, the client socket was closed before the reply was sent. |
| TRANSACTION_ROLLEDBACK | A component participating in the transaction called the rollbackWork transaction primitive to indicate that the transaction should not be committed, or the transaction timed out. You may retry the method calls if desired.<br><br>There is a different cause for this error if you are running EJB CMP entity beans with automatic persistence and timestamp verification (specified on the component properties Persistence tab). If you see "TRANSACTION_ROLLEDBACK: Optimistic Concurrency Control" messages in the server log, try changing the persistence setting to Generated Class, then regenerate stubs and skeletons for the component.<br><br>**Note** In some cases, you may see this message simply because of the activity of other transactions, in which case you must restart the transaction, either in client code or by enabling automatic retry. |
| Trust verification failed. Client certificate not available. | The server listener requires mutual authentication. Make sure the certificate label is set in the client ORB/SSLServiceProvider and that the SSLCallback::getCertificateLabel callback is implemented. |
| Trust verification failed. SSL protocol X.509 certificate chain is incomplete. | Make sure the client's certificate chain is complete. Use the Management Console to verify the client's certificate. If the client's CA certificate is not in the client PKCS token, install it. |

| Message text (or text fragment) | Explanation |
|---|---|
| Trust verification failed. SSL protocol X.509 certificate chain is invalid. | Use the Management Console to verify that the client's certificate chain is valid. |
| Trust verification failed. SSL protocol X.509 certificate has expired. | Use the Management Console to verify the client's certificate. If a certificate has expired, get a new certificate from the CA and install it at the client site. |
| Trust verification failed. SSL protocol X.509 certificate chain contains an unknown CA. | The server administrator must use the Management Console to install the certificate of the CA that signed the client certificate in EAServer, and mark it trusted. |
| Trust verification failed. SSL protocol CA certificate is untrusted. | The server administrator must mark the client's CA as trusted. |
| Unable to initialize the VM | See "Starting the server" in Chapter 3, "Creating and Configuring Servers," in the *EAServer System Administration Guide* for details on selecting and running the Java VM.<br>Check that the PATH and CLASSPATH environment variables are set correctly. |
| Warning: Failed to initialize SSL Service Provider: protocol IIOPS not supported | Verify that the SSL deployment kit is installed and configured correctly and that the JAGUAR_CLIENT_ROOT environment variable points to where the deployment kit is installed.<br>On Windows, verify that the PATH contains *%DJC_HOME%/lib*.<br>On UNIX, make sure the library path variable contains *$DJC_HOME/lib*. |
| Warning: protocol IIOPS not supported | Check the SSL deployment installation, configuration, and the *DJC_HOME* environment variable, as described in the previous warning. |
| Windows Vista error when starting the server | You can safely ignore this error. |

# System exceptions

## EJB components

ejb.logExceptions specifies a global value to enable or disable logging of exceptions thrown by components in an EJB module. If set to true, EAServer logs application and system exceptions that are thrown by the business methods for any component in the EJB module.

To override this setting for individual components, create a <setProperties> command in your user configuration that runs the <logExceptions> subcommand. For example:

```
<target name="configure-user">
  <setProperties component="ejb.components.myjar.MyCompRemote">
     <logExceptions enable="true"/>
  </setProperties>
</target>
```

You can disable logging of exceptions for all components in the server by setting the server Log System Exceptions and Log Application Exception properties. If exception logging is disabled in the server properties, the component settings have no affect.

See Chapter 2, “Deploying and Configuring EJB Components,” in the *EAServer EJB User Guide* for more information about configuring EJB components.

JMetaData exception

If an IIOP client running against EAServer version 6.0 or later gets a JMetaData exception on the server side during lookup, there may be EAServer version 5.*x* classes in the client’s class path. To solve the problem, remove the EAServer 5.*x* classes from the class path. If version 5.*x* classes are required, verify that they are listed in the class path after the version 6.*x* classes. Ideally, there should be no EAServer 5.*x* classes in the class path.

## CORBA system exceptions

The CORBA specification defines the list of standard system exceptions.

CORBA/Java components

In Java, all CORBA system exceptions extend org.omg.CORBA.SystemException. System exceptions are unchecked exceptions (they extend java.lang.RuntimeException). The Java compiler does not require that you catch CORBA system exceptions. However, some exceptions can occur in a well-behaved program. For example, the Session.loookup call throws a NO_PERMISSION exception when you request a component instance and the user lacks permission to instantiate that component. You may want to trap the exceptions shown in the code fragment below:

```
try
{
    // invoke method(s)
    ...
}
```

```
catch (org.omg.CORBA.COMM_FAILURE cf)
{
    // If this occurs when instantiating a Manager
    // instance, the server is likely down or has run
    // out of connections. You can retry the connection
    // if desired.
    //
    // If this occurs after a method call, you
    // can retry the call (or the transaction call
    // sequence for a stateful component).
    ...
}
catch (org.omg.CORBA.TRANSACTION_ROLLEDBACK tr)
{
    // A component on the server aborted the EAServer
    // transaction, or the transaction timed out.
    // Retry the method call(s) if desired.
    ...
}
catch (org.omg.CORBA.OBJECT_NOT_EXIST one)
{
    // Possibly try to create another instance. Check
    // that the package and component are installed
    // on the server.
    // Received when trying to instantiate a component
    // that does not exist. Also received when invoking
    // a method if the object reference has expired
    // (this can happen if the component is stateful
    // and is configured with a finite Instance Timeout
    // property). Create another instance if desired.
    ...
}
catch (org.omg.CORBA.NO_PERMISSSION np)
{
    // Tell the user they are not authorized
    ...
}}
catch (org.omg.CORBA.SystemException se)
{
    // Catch-all clause for any CORBA system exception
    // that was not explicitly caught above.
    // Report the error but don't bother retrying.
    ...
```

**Note** Not all of the possible system exceptions are shown in the example. See the CORBA/IIOP 2.3 specification for a list of all the possible exceptions. You can download the specification from the OMG Web site at http://www.omg.org/technology/documents/idl2x_spec_catalog.htm.

CORBA/C++ components

In C++, all CORBA system exceptions are mapped to a C++ class that is derived from the standard SystemException class defined in the CORBA module. You may want to trap the exceptions shown in this code fragment:

```
try
{
... // invoke methods
}
catch (CORBA::COMM_FAILURE& cf)
{
... // A component aborted the EAServer transaction,
    // or the transaction timed out. Retry the
    // transaction if desired.
}
catch (CORBA::TRANSACTION_ROLLEDBACK& tr)
{
... // possibly retry the transaction
}
catch (CORBA::OBJECT_NOT_EXIST& one)
{
... // Received when trying to instantiate
    // a component that does not exist. Also
    // received when invoking a method if the
    // object reference has expired
    // (this can happen if the component
    // is stateful and is configured with
    // a finite Instance Timeout property).
    // Create a new proxy instance if desired.}
}
catch (CORBA::NO_PERMISSSION& np)
{
... // tell the user they are not authorized
}
catch (CORBA::SystemException& se)
{
... // report the error but don't bother retrying
```

```
}
```

**Note** Not all of the possible system exceptions are shown in the example. See the CORBA/IIOP 2.2 specification (formal/98-02-01) for a list of all the possible exceptions.

CHAPTER 6

# Advanced Topics



## Operational management tools

There are a number of tools and techniques you can use to monitor the operation of the server, its environment, and applications.

Windows platforms provide a number of diagnostic tools:

- See “Windows debugging tools” on page 72.
- Other Windows tools are described under “Runtime monitoring tools” on page 67.

UNIX systems also provide a number of tools:

- See “UNIX debugging tools” on page 73.
- On HP-UX:
  - tusc – to list all system calls; useful for tracing application crashes.
  - vmstat – for virtual memory statistics.
  - top – for a list of processes using the most CPU.

- On AIX, use the System Admin tools.
- On Solaris:
  - top – to see which processes are consuming the most CPU.
  - vmstat – displays CPU and memory usage.
  - pstack – for stack dump analysis.
  - pmap – lists which libraries are loaded and their load locations.
  - pldd – similar to pmap but includes less information.
  - pfiles – lists file descriptors, sockets, files, and so on.
  - truss – displays what is happening with file descriptors, signals, O/S interactions with a process, and so on. truss is useful if you know that a specific action leads to a specific problem.

**Note** truss can generate large volumes of output. To reduce truss output, use truss -p.

## Memory management tools

The following code sample shows how to output information about free memory, total memory, and JVM properties:

```
import java.util.*;
import java.io.*;

public class printMemAndProps {

   public static void main (String[] args) {

   System.out.println(new Date());  // print system date

      // get runtime info and print Free/Total memory
      Runtime rte = Runtime.getRuntime();

      long freeMem = rte.freeMemory();
      long totalMem = rte.totalMemory();

      System.out.println("Free Memory: " + freeMem);
      System.out.println("Total Memory: " + totalMem + "\n");

      java.util.Properties p = null;
```

```
      try {
      p = System.getProperties();
      }
      catch(Exception e) {
      e.printStackTrace();
      return;
      }
      p.list(System.out);  // print all JVM propererties

   }

}
```

### Evaluating Windows memory

You can find a detailed article on Windows memory management titled "Evaluating Memory and Cache Usage" on the Microsoft Web site at http://www.microsoft.com.

This article explains the Performance Console and other Windows tools, with emphasis on how to:

- Monitor memory
- Interpret output
- Identify memory leaks—see "Investigating User-Mode Memory Leaks" and subsequent sections

### PowerBuilder memory tuning

For information about tuning and troubleshooting the C/C++ heap memory manager that is used by EAServer and PowerBuilder, see EAServer/PowerBuilder Memory Tuning and Troubleshooting at http://www.sybase.com/detail?id=1027319.

## Runtime monitoring tools

The tools described in this section enable you to perform EAServer runtime monitoring.

## Management Console

In the Management Console, HTTP and IIOP network monitoring shows thread usage, and IIOP network monitoring shows the host and thread information for current clients. For details, see Chapter 11, “Runtime Monitoring,” in the *EAServer System Administration Guide*.

See “EAServer tracing” on page 9 for a description of the properties that you can configure EAServer to trace.

## EAServer monitoring APIs

The EAServer monitoring APIs allow you to monitor aspects of your runtime environment such as sessions, threads, components, connection caches, listeners, IIOP and HTTP traffic, and so on. Available information includes:

- Peaks
- Maximums
- Current values
- Forced connects

Using the Jaguar::Monitoring API, you can:

- Write service components that periodically check desired characteristics such as peak values and forced connections
- Retrieve configured values for maximum HTTP threads and the maximum number of simultaneous client threads, as well as the existing keys for current thread usage
- Return information on IIOP clients using the getConnectedUsers method

Jaguar::PerfMonitor is a performance monitoring interface that provides performance statistics in a per-second, per-minute, and per-hour bucket model.

Jaguar::StatProvider and Jaguar::StatProviderController are interfaces implemented by *statistic provider* components that collect performance statistics. EAServer includes statistics providers for the connection caching and HTTP protocol handler subsystems. You can also implement your own statistics providers using these interfaces.

For information on the monitoring APIs, see the generated HTML documentation for the Jaguar IDL module, in this file within your installation:

```
html/ir/Jaguar.html
```

## jagtool

The jagtool commands getserverinfo and getserverstate allow you to get additional information about the server status. getserverinfo returns the server status and version number. getserverstate queries the state of service components and is useful in scripts that start or restart servers; use it to determine whether the server is ready to accept client connections by checking whether the name service status is "STOPPED." Custom services can implement an additional method, getServiceState, to allow jagtool to query their status. For more information, see the *EAServer Automated Configuration Guide*:

- Chapter 4, "Creating Service Components"
- Chapter 6, "Using jagtool and jagant"

## Service components

You can use service components to perform background processing or to provide common services for EAServer clients and other EAServer components.

To add service components to EAServer when the server is not running, which may be required by OEM clients, replace the EAServer property file with a property file that has the appropriate services listed for start-up by doing one of:

- Replace *default.properties*, located in *Repository/Instance/com/sybase/djc/server/ApplicationServer*, with an appropriate properties file before starting the server for the first time, or,
- Create a configuration script, or modify *default-application-servers.xml*, and run the configure command before starting the server for the first time, or,
- Update <*ServerName*>*.properties*, either by using a configuration script or by replacing it with a preconfigured properties file.

## ListDLLs

ListDLLs is available on the Sysinternals Web site at http://www.sysinternals.com. It is a Windows tool that displays all the DLLs that are currently loaded, including load locations and their version numbers, for each process. Version 2.0 prints the full path names of loaded modules. This tool can be very useful for verifying whether the correct DLL versions are loaded in a process.

Run ListDLLs in a DOS window at the command line to show all processes. For best results, redirect the output to a file so that you can review the file at your convenience. To redirect the output to a file named *myfile.txt*, which you can open in Notepad or any text editor, enter:

```
listdlls start-server >myfile.txt
```

## Process Explorer

Process Explorer is a GUI version of ListDLLs that displays DLLs and handles that are in use by a specific process. Process Explorer is available on the Sysinternals Web site at http://www.sysinternals.com. Full information is available in the *Readme* file included in the downloaded file.

After you download and unzip the program file (*procexp.exe*):

1 Select Start | Run. Enter the full path to *procexp.exe*.

2 Select View DLLs.

3 In the Process view in the top pane, select the *java.exe* process to see which DLLs the process has loaded.

You can also use the Process Explorer to:

- Save the output to a file (File | Save and File | Save As)
- View the DLL or handle properties
- Show the process tree, including parent-child relationships
- Refresh the view
- Set various options
- Search
- Display online help

## Profiling tools

Profiling tools allow you to identify and track performance issues.

For Java profiling, refer to the following documents:

- White paper #1011357: Integrating Optimizeit in Sybase EAServer at http://www.sybase.com/detail?id=1011357

  This document shows how you can integrate the Optimizeit Java profiler with EAServer.

- White paper #1011550: Memory Management within Java Processes at http://www.sybase.com/detail?id=1011550

  This document assists in monitoring and managing memory in Java processes.

For C++ profiling, use Purify (UNIX) or PurifyPlus (Windows) from Rational. Purify is designed to track down memory leaks and invalid memory-use errors.

For more information, check Rational documentation at http://www.rational.com/support/documentation/manuals/index.jsp.

### ps, pstat, pmon, and PsList

The ps command in UNIX systems displays information about active processes.

On Windows, you can use pstat and pmon for similar information. PsList, which is freeware available from the Sysinternals Web site at http://www.sysinternals.com, combines much of this information so you can view processes, CPU and memory information, or thread statistics. PsList can provide either summary or detailed data.

## Debugging tools

Debuggers and crash handlers can:

- Generate and analyze stack dumps and core files (see “Stack traces, dump files, and core files” on page 74)
- Attach to a running process
- Step through and debug specific components

### Java debugging tools

For Java, you can use jdb, a simple command line debugger for Java classes, or other tools that support the same remote debugging interfaces as EAServer.

EAServer supports the Java Platform Debugger Architecture (JPDA). To run the server in JPDA mode, use the -jpda option when you start the server; for example, on UNIX:

```
start-server.sh  -jpda [-jpdaSuspend]
```

Used with jpda, jpdaSuspend 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. For more information about command line options you can use when starting the server, see "Starting the server," in Chapter 3, "Creating and Configuring Servers," in the *EAServer System Administration Guide*.

For details about Java debugging, see:

- Chapter 12, "Developing CORBA/Java Components," in the *EAServer CORBA Components Guide*, and
- The Sun Developer Network Web site at http://java.sun.com.

## Windows debugging tools

Windows provides several debugging options, which may require help from Sybase Technical Support to use effectively:

- AutoDump+, also known as ADPlus, a console-based tool to help troubleshoot a process or application that stops responding or fails.
- CDB, a console program for debugging kernel-mode drivers on Windows NT.
- Dr. Watson, which records program errors in a log file for analysis. Configuration options let you define:
  - The type of files to output and where those files are located
  - Options to dump all thread contexts and symbols
  - The number of errors or instructions to save

  For details:
  - Run *drwtsn32.exe*, then click Help.
  - Go to the Microsoft Web site at http://www.microsoft.com for information on how to read a Dr. Watson log.
- Kernel debugger (*kd.exe*), a command line debugger.

  Use the kd utility to load a dump file or attach to a running process before a crash occurs. You can view memory or the callstack, go to a specific offset, and so on.

- UserDump, which generates a user dump of a process that shuts down with an exception or hangs. UserDump (*userdump.exe*) is included in the Microsoft OEM Support Tools Package; set it up from the Control Panel.

  **Note** UserDump and Dr. Watson differ in the types of exceptions they track. If your server inexplicably disappears, you may find UserDump useful for analysis.

- WinDbg, an analysis tool that Technical Support can use to examine the *.dmp* file generated by the UserDump utility. Accurate analysis of the *.dmp* file requires access to the EAServer and PowerBuilder symbol files, which customers do not have.

Table 6-1 summarizes the features of Windows dump analysis tools:

***Table 6-1: Windows Dump Analysis Tools***

| Able to create dump file upon | AutoDump+ | CDB | Dr. Watson | UserDump |
|---|---|---|---|---|
| Crash | Yes | Yes | Yes | Yes |
| Exception | Yes | Yes | Yes | Yes |
| Hang | Yes | Yes | No | Yes |
| Start-up failure | No | Yes | No | Yes |
| Normal run | No | Yes | No | No |

See the Microsoft Web site at http://www.microsoft.com for more information about these debugging tools.

## UNIX debugging tools

Common UNIX debuggers include:

- dbx, which works with programs compiled with debugging information (usually by compiling with the -g option). You can also probe core dumps with dbx to determine the cause of the crash.

  dbx is available on Solaris and Digital UNIX.

- gdb, the gnu debugger, which is a command line debugger that can debug programs compiled on a variety of different compilers (including C and Fortran), on several platforms.

## Attaching a debugger to EAServer

You can attach a debugger to the jagsrv process. To do this, run WinDbg and attach to process jagsrv before a crash occurs. Use symbol files to see the call stack, memory, and so on.

If EAServer is running as a service, you may be able to attach the debugger to the Windows service. More information is available on the Microsoft support site at http://support.microsoft.com.

# Stack traces, dump files, and core files

Stack traces, dump files, and core files contain useful information about what a server process is doing at a given time, such as:

- Methods called
- Memory information
- Active thread information

These files are time-consuming to read and not always easy to understand. Try simple troubleshooting techniques first. Use the server log, which is much more readable, to review server and component output and check any errors raised. See “Logging and statistics” on page 4 for more information on log files. Next, check the dump file to get an idea of which debug/trace flags should be turned on. This may help identify things like operating system signal issues.

This section explains how to obtain dump and core files for troubleshooting. See “Using stack traces” on page 8 for a complete list of available traces and how to obtain a trace.

---

**Note** Stack traces, dump files, and core files are not mutually exclusive. Depending on your platform, and the tools and options you use, the output file may contain different types of data.

---

## Windows dump files

When an unhandled exception or fault occurs, Windows generates either a *.dmp* or a *.log* file based on system settings.

The *.log* file:

- Is generated by Dr. Watson (see "Windows debugging tools" on page 72)
- Is readable with any text editor
- Includes details about the environment, active programs, services, and modules (DLLs) when the crash occurred

A *.dmp* file is generated by tools like the kernel debugger, UserDump, and Dr. Watson if you select the Create Crash Dump option. It includes dumps of memory, call stack, offsets, and so on. A log file is also generated.

You can use various tools to read the dump file. For a comparison of the kinds of dump file analysis provided by the tools, see "Windows debugging tools" on page 72.

A Windows Registry entry determines which program handles uncaught exceptions.

For information on how to configure your preferences, go to the Microsoft support site at http://support.microsoft.com.

**Note** For help interpreting Windows *.dmp* or *.log* files, contact Sybase Technical Support.

## UNIX core files

The core file contains memory values and a stack trace for a running process. Core files are generally usable only on the same machine where the server is running and the core file was generated. The syntax to create a core file is:

```
gcore [option] process_id
```

You can use the resulting core image with debugging utilities such as sdb, adb, or dbx—see "UNIX debugging tools" on page 73.

You can create core files only on Solaris, HP-UX, and Linux; you cannot on AIX.

# Class loader configuration issues

This section presents information that can be useful when diagnosing class loader configuration problems.

## Common problems with custom class lists

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 can fail, and you may 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 calling entity and the caller 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. For more information, see Chapter 10, “Configuring Java Class Loaders,” in the *EAServer System Administration Guide*.
  - 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** You must refresh classes 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.

## Custom class loader tracing

To trace a class loader, use these options when you start EAServer, where *NamedClassLoader* is the name of the class loader and *NamedClassLoaderLog* is the name of the log file:

```
-Ddjc.trace=com.sybase.djc.util.NamedClassLoader
-Ddjc.trace=com.sybase.djc.util.NamedClassLoaderLog
```

To investigate class loader problems, see "Troubleshooting class loader configuration issues" in Chapter 10, "Configuring Java Class Loaders," in the *EAServer System Administration Guide*.

## 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 the JAR file while the server is running.

To allow refresh of custom-loaded JAR files, 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.

# Troubleshooting Web services

This section describes how to determine why an EAServer Web service is inaccessible or working improperly.

For information about using EAServer Web services, see Chapter 1, "Overview of Web Services in EAServer," in the *Web Services Toolkit User's Guide*.

## Check logs and error messages

To determine the cause of a Web service problem, check *<server-name>.log*, which is located in the EAServer *logs* subdirectory.

Table 6-2 describes common Web services error messages.

***Table 6-2: Web services error messages***

| Error message | Description |
|---|---|
| Error 500 Servlet jspservlet: unable to service request: Provider org.apache.xerces.jaxp.DocumentBuilderFactoryImpl not found. | Occurs when the *xerces.jar* file is found in neither the Web application's class path nor in the EAServer */java/lib* subdirectory. |
| AxisFault faultCode: Server.userException faultSubcode: faultString: No such operation. | Verify that you are using the correct datatypes for the parameter values. |
| WSTAdminException: com.sybase.wst.admin.WSTAdminException:java.lang.ClassCastException: org.apache.crimson.jaxp.DocumentBuilderFactoryImpl | This indicates a class loader issue, which can occur when you deploy a Web service if you have specified a dependency class in the deployment descriptor, and the same class was loaded by the EAServer system class loader.<br>To work around this problem, do not include dependency JAR files for third-party libraries to the Web application if those libraries are already provided by EAServer and loaded during start-up. |

## Verify WSDL files and SOAP addresses

For a specific collection, EAServer allows you to generate a list of its Web services, links to their corresponding Web Services Description Language (WSDL) files, and a list of the operations in each service. To display this information, open a Web browser, and access the following URL, where *host* represents the machine name where EAServer is running, *http_port* is the port number of the EAServer HTTP listener, and *collection* is the name of the Web service collection:

```
http://host:http_port/collection/services
```

For example, the following URL:

```
http://localhost:8000/ws/services
```

could generate the output below for the *ws* service, which includes two Web services, *n_pbhello* and *GoogleSearchPort*:

```
n_pbhello (WSDL link)
  o fhello
```

```
GoogleSearchPort (WSDL link)
  o doGetCachedPage
  o doSpellingSuggestion
  o doGoogleSearch
```

To verify a WSDL file, click the WSDL link. The contents of the WSDL file should display in the browser. You can also open a WSDL file in EAServer, using the following URL, where *host* represents the machine name where EAServer is running, *port* is the port number of the EAServer HTTP listener, *collection* is the name of the Web service collection, and *service_name* is the name of the Web service:

```
http://host:port/collection/services/service_name?wsdl
```

For example, the following URL opens the WSDL file for the *n_pbhello* Web service in the *ws* collection:

```
http://localhost:8000/ws/services/n_pbhello?wsdl
```

To verify that a Simple Object Access Protocol (SOAP) address is accessible:

1 Determine the URL for the Web service using the EAServer wstool utility as follows, where *collection* is the name of the Web service collection and *service_name* is the name of the Web service:

```
wstool list URL service:collection/service_name
```

2 In a Web browser, access the Web service's URL; for example:

```
http://localhost:8000/ws/services/n_pbhello
```

Accessing the SOAP address does not invoke the Web service, but should display a message similar to the following:

```
n_pbhello
Hi there, this is an AXIS service!
```

## Invoke operations and create a test client

Using the Web Services Eclipse plug-in, you can invoke a service's operations:

1 In Eclipse, expand the collection (*ws* is the default), then expand the Operations folder under the service name.

2 Right-click the operation, and choose Invoke from the menu.

3 Specify the input values, if required for the operation, then click Invoke. The results display.

To create a test client, use either of these tools; both are described in the *Web Services Toolkit User's Guide*:

- Management Console—see Chapter 5, "Management Console—Web Services," or
- wstool utility—see Chapter 9, "Using wstool and wstant."

## View incoming and outgoing SOAP messages

EAServer includes the Apache *soap.jar* file. This JAR file includes the TcpTunnel and TcpTunnelGui utility classes, which can be used to proxy HTTP requests. These classes allow you to view all HTTP request headers, reply headers, and content for incoming and outgoing SOAP messages. To use the TcpTunnelGui utility:

1 Start EAServer.

2 Run the following command, where *tunnel_port* is an unused port to which proxy requests can be directed, *server* is the name of the machine where EAServer is running, and *http_port* is the EAServer HTTP listener port:

```
java org.apache.soap.util.net.TcpTunnelGui tunnel_port server http_port
```

3 Invoke the Web service operation using the *tunnel_port* number, instead of the EAServer HTTP port number.

SOAP Inspector view

To view SOAP messages in the SOAP Inspector view:

1 Open the SOAP Inspector view.

2 Enable messages.

3 Run the Web Service client as a Web Services application.

## PowerBuilder Web service considerations

- If a Web service implementation is a PowerBuilder component, verify that you can invoke the corresponding method on the underlying PowerBuilder component from a PowerBuilder client program.

- If you invoke an EAServer Web service from a PowerBuilder client, verify that the client program is not loading an old copy of the *libeay32.dll* (for example, from *WINNT\system32*), because this can cause the PowerBuilder Web service invocation to fail.

# EAServer plug-in for JBuilder

This section describes how to troubleshoot problems associated with the EAServer plug-in for JBuilder. For information about installing the plug-in, see white paper #1028173: Configuring and Troubleshooting the Sybase EAServer Plug-in for JBuilder 2005 at http://www.sybase.com/detail?id=1028173.

❖ **Verifying your configuration**

If you are having trouble with the EAServer plug-in for JBuilder, verify:

1 You have the latest *easerver-jbsp.jar* file.

2 The working directory matches the EAServer installation location.

3 The plug-in is set up using the JBuilder Project | Default Project Properties.

4 The project's application server is set to EAServer. Check the JBuilder Project | Project Properties | Servers menu item.

5 If you are debugging a JSP, you are using a local EAServer installation.

6 If the WAR file contains extra files, or if it contains other application servers, such as the BNX Authentication Suite (BAS) 5.0:

   a Right-click the Web application, and choose the Dependencies tab.

   b Select EAServer, and select Never Include any Classes or Resources.

❖ **Reviewing deployment output and generating verbose logging**

1 Examine the output that was generated during deployment. Pay special attention to the class path.

2 Display extra internal logging information and runtime exceptions by running JBuilder with the -verbose option; for example:

```
jbuilder\bin\jbuilder -verbose > output.txt
```

where *output.txt* is the log file.

3 Examine the EAServer log files, which are typically found in the EAServer *bin* or *devbin* subdirectory.

❖ **Deploying externally**

1 Deploy the WAR file to EAServer using the Management Console or the deploy command line tool.

2 Run the J2SDKEE Sun verifier on the generated EAR, JAR, or WAR file; for example, on Windows:

```
verifier.bat myWAR.war
```

## JBuilder JSPs and ResultSets

This section describes how to troubleshoot problems associated with JSPs in JBuilder that retrieve ResultSets using EAServer connection caches. For information about creating JSPs in JBuilder that perform this task, see white paper #1028828: Create JSP in JBuilder to Retrieve ResultSet from EAServer Connection Cache at http://www.sybase.com/detail?id=1028828.

❖ **Troubleshooting JSPs in JBuilder**

1 Verify your configuration—see “Verifying your configuration” on page 81.

2 If the JSP does not compile, review the content of the JBuilder Messages pane for information about why the compilation failed.

3 If attempts to deploy the Web module fail:

a Review the content of the JBuilder Messages pane for information about why the deployment failed.

b Verify that EAServer is running.

c Verify that the server name and port number in the project deployment settings match the actual running server and HTTP listener.

4 If the Web browser displays a “page not found” error:

a Verify that EAServer is running.

b Verify that the HTTP listener can process requests by attempting to access the default documentation page; for example, http://*serverName*:8000.

c Restart EAServer.

5 If the Web browser displays an exception, or a message that indicates there is a problem with the page; for example, “internal server error 500”:

a Check the EAServer log files *serverName.log* and *serverName-http-YYYY-MM-DD.log* for details, where:

- *serverName* is the name of the server, and
- *YYYY-MM-DD* is the current date.

b Add an error page to the JSP, then rebuild and redeploy the project. For example:

1 Add the following line to your JSP:

```
<% page errorPage=”jsp2_error.jsp” %>
```

2 Create a new JSP named *jsp2_error.jsp* with the following content:

```
<%@ page isErrorPage="true" %>
<html>
<body>
<h1>Error page</h1>
<br>Error occurred in the JSP: <%= exception.getMessage() %>
<Br>Stack Trace:
<%
java.io.CharArrayWriter cw = new java.io.CharArrayWriter();
java.io.PrintWriter pw = new java.io.PrintWriter(cw,true);
exception.printStackTrace(pw);
out.println(cw.toString());
%>
</body>
</html>
```

6 If you make any changes to the Web application, rebuild the WAR file before redeploying it to EAServer; otherwise, the WAR file does not include your changes.

# WINS and server response time

If you are using a Microsoft Windows Internet Naming Service (WINS) server, response times can be improved by configuring your system to allow NetBIOS traffic to and from EAServer.

## Windows XP and Service Pack 2

If you are running Windows XP and have Service Pack 2 installed, establishing a session between a client and a server may take up to five seconds. This problem exists because the Windows XP firewall blocks NetBIOS traffic between workstations and EAServer. To work around this issue, configure the XP firewall to allow traffic on port 137 to and from EAServer.

## Cisco VPN clients

If a Cisco VPN client is installed, the Cisco stateful firewall is deployed, which does not allow NetBIOS traffic, and thus slows server response time. To permit NetBIOS traffic, set the VPN client status to Connected.

## Personal firewalls and router ACLs

Personal firewalls, such as Zone Alarm, block NetBIOS traffic. If a personal firewall or Web-database (WDB) controlled hardware firewall is deployed between the WDB network and the internal network, it must be configured to allow NetBIOS traffic (on port 137) to and from EAServer.

Routers with access control lists (ACLs) may also block NetBIOS traffic. If a WDB-controlled router is running ACLs, it must be configured to allow NetBIOS traffic (on port 137) to and from EAServer.

# Installing and compiling Apache on HP RISC

Installing and compiling Apache on the HP RISC platform can be problematic, due to defects in Apache build scripts and problems linking to Apache libraries.

---

**Note** The following instructions are guidelines for building with SSL. You may need to tweak these steps, based on your environment and machine configuration.

---

❖ **Building Apache on HP RISC to use EAServer plug-in libraries**

1. Compile Apache using aCC. By default, Apache searches for both cc and aCC, but use aCC to compile all code, including the C code in Apache. Also use aCC to link the httpd binary, because aCC links libCsup, libstream, and libstd, which are necessary to load C++ libraries. The EAServer redirector plug-ins are all written in C++.
2. Use the GNU make tool gmake to build Apache. Apache uses GNU autconf and automake to generate makefiles.
3. Install Perl version 5.0 or later on your machine. You can download the Perl binary *depot* file from the HP Web site at http://www.hp.com. You need Perl version 5.0 or later to build OpenSSL.
4. Download OpenSSL to your machine, and follow the instructions in the README and INSTALL files. You can download OpenSSL from the OpenSSL Web site at http://www.openssl.org.
5. Download the Apache 2.0 *.tar* file, and extract the contents into the */work/httpd-2.0.53* directory. You can download Apache from one of the mirror sites listed on the Apache Software Foundation Web site at http://www.apache.org/dyn/closer.cgi.
6. Change to the */work/httpd-2.0.53* directory, and run:

```
CC="aCC" CFLAGS="-Ae +Z +DA1.1 +DS2.0 +u4 -D_HPUX -DHPUX \
-D_POSIX_C_SOURCE=199506L -D_HPUX_SOURCE -DNATIVE" \
LDFLAGS="-v -Wl,-v,+s,+n"  LD="aCC" \
./configure --prefix=/work/apache2 \
--enable-mods-shared=all --enable-rewrite=shared \
--enable-speling=shared --disable-auth-digest \
--enable-ssl --with-ssl=/work/ssl
```

Note the name of the module that reports problems during configuration. You can choose to disable this module.

Verify that there are no errors reported during configuration. If you ignore configuration errors, Apache will be installed correctly. Configuration requires several minutes to complete. If it completes too quickly, it probably failed.

OpenSSL is installed in the */work/ssl* directory, and includes *bin*, *include*, and *lib* subdirectories.

7. Run gmake, and capture the console output to help debug errors:

```
gmake 2>&1 | tee out
```

**Note** This command works in Korn shell and Bash. It does not work in C shell.

Errors may occur when building apr and apr-util/xml/expat. The errors usually correspond to linking libraries, because of incorrect link commands. Apache may not be tested to run an entire build with aCC, so it is necessary to work around bugs in their makefiles.

To debug build errors associated with apr and apr-util/xml/expat, use the following example as a guide. You can decide how to fix the problem based on the error messages you receive.

a To debug apr, change to */work/httpd-2.0.53/srclib/apr/build*.

b Using a text editor, open *libtool.m4*, and search for "+h" and "+b." Verify that each command line with a "+h" or "+b" looks similar to:

```
_LT_AC_TAGVAR(archive_cmds, $1)='$CC -b ${wl}+h ${wl}$soname -
Wl,+b,$install_libdir -o $lib $predep_objects $libobjs $deplibs
$postdep_objects $compil er_flags
```

Preceding each "+h" or "+b" and its value must be either "${wl}" or "-Wl." In the example above:

- `${wl}+h ${wl}$soname`
- `Wl,+b,$install_libdir`

c For apr, change to */work/httpd-2.0.53/srclib/apr*

For apr-util/xml/expat, change to */work/httpd-2.0.53/srclib/apr-util/xml/expat*.

d Using a text editor, open *libtool*, and verify the syntax for the "+h" and "+b" strings, as described in step b.

```
archive_cmds="\$LD -b \${wl}+h \${wl}\$soname \${wl}+b
\${wl}\$install_libdir -o \$lib \$libobjs \$deplibs \$linker_flags"
```

e Check the build settings in the **.mk* files.

f Change to */work/httpd-2.0.53*, and run gmake again.

g To determine which command line is being executed, disable the LTFLAGS variable. This variable sets libtools to silent. Use grep to search for "silent" in the *libtool* or *libtool.m4* directory, then comment out this variable.

h Once all build errors are resolved, verify that the *httpd* binary is linked correctly:

1 Delete */work/httpd-2.0.53/httpd*.

2 Change to */work/httpd-2.0.53/build*.

3 Using a text editor, open *config_vars.mk*. Search for these variables, and comment them out:

- CFLAGS
- EXTRA_CPPFLAGS
- LTFLAGS (comment out to display entire command line)

4 Change to */work/httpd-2.0.53*, and re-run gmake.

5 When the build is error free, change to */work/httpd-2.0.53/.libs*, and run:

```
chatr httpd
```

This should show you all the libraries that are linked to *httpd*. Verify that *libCsup*, *libstream*, and *libcl* are linked. If they are not, there is a problem.

6 Once you verify that there are no errors, change to */work/httpd-2.0.53*, and run:

```
gmake install
```

The Apache files should be installed to */work/apache2*, based on your prefix.

8 Edit */work/apache2/bin/envvars*, and set SHLIB_PATH to the location of the JDK libraries, and set JAGUAR_CLIENT_ROOT to the EAServer installation directory (required for SSL):

```
SHLIB_PATH="/work/apache2/lib:/sun/jdk/jdk1.3.1_10/jre/lib/PA_RISC2.0:/
sun/jdk/jdk1.3.1_10/jre/lib/PA_RISC2.0/native_threads:/sun/jdk/jdk1.3.1
_10/jre/lib/PA_RISC2.0/classic:$SHLIB_PATH"
export SHLIB_PATH

JAGUAR_CLIENT_ROOT=/Sybase/EAServer
export JAGUAR_CLIENT_ROOT
```

Now, you should be able to start an Apache 2.0 server with an EAServer compatible configuration.

# Miscellaneous topics

## Testing and debugging classes

If Sybase provides classes for testing and debugging at your customer site, put the classes in the *DJC_HOME/lib/patches* directory. Put only class in this directory, not JAR files. Verify that class files are unzipped and in the correct packages.

## Additional tools and utilities

For more information about tuning and debugging tools and utilities, see the Monitor/Tune folder on the EAServer CodeXchange Web page at http://easerver.codexchange.sybase.com/servlets/ProjectDocumentList.

## Internet Explorer security patch

If you install Internet Explorer Security Patch MS01-055, Web applications that use session cookies may not work. This security patch denies cookies from servers whose domain names do not comply with the specifications of RFC 833; for example, names that include underscores are not supported. The result is that session variables may not be maintained when dealing with some Web applications. For more information, see the Microsoft Knowledge Base article at http://support.microsoft.com/kb/312461/en-us.

## Drivers that use the DataSource interface

Drivers implementing the DataSource interface are treated differently than "simple" drivers. This is the difference between JDBC level 1 and JDBC level 2 (also known as "JDBC specification version 2.0"). Applications must use getXXX and setXXX methods to pass the user name, password, and other information to the driver. The get/set methods you use are URL-dependent. For example, if you connect to an Oracle database using an Oracle JDBC driver, and your Oracle cache URL is as listed below, EAServer calls these methods on the driver instance: setDriverType(thin), setServer(conlabtt), setPort(1521).

```
DriverType=thin:Server=conlabtt:Port=1521:DatabaseName=conlabtt
```

To find the exact name of the driver class, and the properties, run:

```
$JAVA_HOME/bin/javap oracle.jdbc.pool.OracleDataSource
```

## Java Message Service

The EAServer message service allows you to send or publish messages to a queue or topic, where they are stored until they can be delivered to either a client or a component. See the Java Message Service User's Guide at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc00486_0600/html/easjms/title.htm.

### JMS and jagtool

If you are using jagtool, you may not see all the EAServer message queues and topics. The EAServer 5.*x* message service works differently than the EAServer 6.*x* message service. In EAServer 5.*x*, the message service is based on EAServer CTS components, while the EAServer 6.*x* message service uses the JMS APIs. Using jagtool, the only message queues and topics that are visible are the ones that were created in EAServer 5.*x* using the CTS APIs.

To see all the message queues and topics, use the Management Console. For new development, use the Management Console to create message queues and topics.

### Alert Management System and the message service

The Alert Management System (AMS) version 4.1.1 processes and manages XML alert messages. The main component of AMS is an MDB (message-driven bean) that writes alert message to a JMS topic. If you run this application with EAServer, both the MDB and the EAServer message service must use the same connection cache; otherwise, errors similar to the following are written to *serverName.log*:

```
JCM Caught Throwable: java.sql.SQLException: Error: Current enlistment requires
2PC Resource and No 2PC Resource Configured for Cache:SybaseJMS
```

# Index

## Numerics



## A



## B



## C

## D

## E



## F



## G

## H



## I



## J

## K



## L



## M

## N



## O



## P

## R



## S

## T



## U



## V

## W



## X