Troubleshooting Guide

EAServer
5.0
## Contents

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

### CHAPTER 1 Monitoring Techniques ................................................................. 1

- Overview .......................................................................................................................... 1
- Gathering information .................................................................................................... 1
- Logging ............................................................................................................................ 4
  - EAServer log ........................................................................................................ 4
  - HTTP logs ........................................................................................................ 6
  - IIOP log ................................................................................................................ 7
  - Installation log .................................................................................................... 8
  - Other useful data ................................................................................................. 8
- Using stack traces ........................................................................................................ 8
  - Obtaining stack traces ...................................................................................... 9
  - EAServer traces .............................................................................................. 10
  - Java exception traces ..................................................................................... 15
- Tracing network problems .......................................................................................... 17
  - TCP/IP .............................................................................................................. 17
  - IIOP ................................................................................................................ 17
  - HTTP .............................................................................................................. 17
  - TDS .................................................................................................................. 18

### CHAPTER 2 Common Problem Areas .............................................................. 19

- Installation issues ....................................................................................................... 19
- Server crashes, hangs, or disappears ......................................................................... 20
  - Server crashes ................................................................................................. 20
  - Server hangs ..................................................................................................... 24
  - Server disappears ............................................................................................. 26
- Server slows or runs out of memory .......................................................................... 27
- Server starts in Admin mode ..................................................................................... 27
- Connection problems ............................................................................................... 28
- Application issues ..................................................................................................... 28
  - Generic issues .................................................................................................. 28
  - Java component issues ..................................................................................... 29

Troubleshooting Guide
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction .................................................................................. 55</td>
</tr>
<tr>
<td>Error messages ............................................................................. 56</td>
</tr>
<tr>
<td>System exceptions ......................................................................... 64</td>
</tr>
<tr>
<td>CHAPTER 6 Advanced Topics ................................................................ 67</td>
</tr>
<tr>
<td>Operational management tools ...................................................... 67</td>
</tr>
<tr>
<td>Memory management tools ................................................................ 68</td>
</tr>
<tr>
<td>Runtime monitoring tools ................................................................ 69</td>
</tr>
<tr>
<td>Debuggers ....................................................................................... 74</td>
</tr>
<tr>
<td>Java debugging tools .................................................................... 74</td>
</tr>
<tr>
<td>Windows debugging tools ................................................................ 74</td>
</tr>
<tr>
<td>UNIX debugging tools ................................................................... 76</td>
</tr>
<tr>
<td>Attaching a debugger to EAServer ................................................. 76</td>
</tr>
<tr>
<td>Stack traces, dump files, and core files ...................................... 76</td>
</tr>
<tr>
<td>Windows dump files ....................................................................... 77</td>
</tr>
<tr>
<td>UNIX core files ............................................................................ 78</td>
</tr>
<tr>
<td>EAServer plug-in for JBuilder ...................................................... 78</td>
</tr>
<tr>
<td>JBuilder JSPs that use connection caches to retrieve ResultSets ... 79</td>
</tr>
<tr>
<td>Miscellaneous topics ...................................................................... 81</td>
</tr>
<tr>
<td>Additional tools and utilities ....................................................... 81</td>
</tr>
<tr>
<td>Internet Explorer security patch ................................................. 81</td>
</tr>
<tr>
<td>Drivers that use the DataSource interface ..................................... 81</td>
</tr>
<tr>
<td>Systems Management Web console ................................................. 82</td>
</tr>
<tr>
<td>Alert Management System and the message service ..................... 82</td>
</tr>
<tr>
<td>Duplicate server names in EAServer Manager .............................. 82</td>
</tr>
<tr>
<td>Index ................................................................................................ 83</td>
</tr>
</tbody>
</table>
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 5.0, 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 to help resolve your problem more quickly.
- A greater understanding of EAServer.

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

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.

The core EAServer documents are available in HTML format in your EAServer software installation, and in PDF and DynaText format on the Technical Library CD.
What’s New in EAServer summarizes new functionality in this version.

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

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

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

The EAServer Programmer’s Guide explains how to:
- Create, deploy, and configure components and component-based applications
- Create, deploy, and configure Web applications, Java servlets, and JavaServer Pages
- Use the industry-standard CORBA and Java APIs supported by EAServer

The EAServer Web Services Toolkit User’s Guide describes Web services support in EAServer, including:
- Support for standard Web services protocols such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Uniform Description, Discovery, and Integration (UDDI)
- Administration tools for deployment and creation of new Web services, WSDL document creation, UDDI registration, and SOAP management

The EAServer Security Administration and Programming Guide explains how to:
- Understand the EAServer security architecture
- Configure role-based security for components and Web applications
Configure SSL certificate-based security for client connections using the EAServer Manager plug-in for Sybase Central

* Implement custom security services for authentication, authorization, and role membership evaluation

* Implement secure HTTP and IIOP client applications

* Deploy client applications that connect through Internet proxies and firewalls

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


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

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


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

The formatting conventions used in this manual are:

### Formatting example

<table>
<thead>
<tr>
<th>To indicate</th>
<th>commands and methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>When used in descriptive text, this font indicates keywords such as:</td>
<td></td>
</tr>
<tr>
<td>• Command names used in descriptive text</td>
<td></td>
</tr>
<tr>
<td>• C++ and Java method or class names used in descriptive text</td>
<td></td>
</tr>
<tr>
<td>• Java package names used in descriptive text</td>
<td></td>
</tr>
<tr>
<td>• Property names in the raw format, as when using jagtool to configure applications rather than EAServer Manager</td>
<td></td>
</tr>
</tbody>
</table>

### variable, package, or component

<table>
<thead>
<tr>
<th>Italic font indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Program variables, such as myCounter</td>
</tr>
<tr>
<td>• Parts of input text that must be substituted, for example:</td>
</tr>
<tr>
<td>Server.log</td>
</tr>
<tr>
<td>• File names</td>
</tr>
<tr>
<td>• Names of components, EAServer packages, and other entities that are registered in the EAServer naming service</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other sources of information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the Sybase Getting Started CD, the Sybase Technical Library CD and the Technical Library Product Manuals Web site to learn more about your product:</td>
</tr>
<tr>
<td>• The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the Technical Library CD. It is included with your software. To read or print documents on the Getting Started CD you need Adobe Acrobat Reader (downloadable at no charge from the Adobe Web site, using a link provided on the CD).</td>
</tr>
<tr>
<td>• The Technical Library CD contains product manuals and is included with your software. The DynaText reader (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.</td>
</tr>
<tr>
<td>Refer to the Technical Library Installation Guide in your documentation package for instructions on installing and starting the Technical Library.</td>
</tr>
</tbody>
</table>

---
• The Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Updates, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

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

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ Finding the latest information on product certifications

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2 Select Products from the navigation bar on the left.
3 Select a product name from the product list and click Go.
4 Select the Certification Report filter, specify a time frame, and click Go.
5 Click a Certification Report title to display the report.

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

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

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ Finding the latest information on EBFs and software maintenance

2 Select EBFs/Maintenance. Enter user name and password information, if prompted (for existing Web accounts) or create a new account (a free service).
3 Select a product.
4 Specify a time frame and click Go.
Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

EAServer 5.0 and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

The online help for this product is also provided in HTML, which you can navigate using a screen reader.

For information on using this product without a mouse, see “Keyboard navigation” in Chapter 2, “Sybase Central Overview,” of the EAServer System Administration Guide.

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

For 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.

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 may require that you restart the server for changes to take effect, or to test the result and impact.

Gathering information

When you are investigating an EAServer error, before you report the problem to Sybase Technical Support, gather the following information to expedite a solution:

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

2  Information about your EAServer installation:
Gathering information

- Version number
- Build number
- Edition

This information appears at the start of the server log. See “Logging” 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 71. To list which libraries are loaded and their load locations, use pmap, or a similar command for your platform.

4 Relevant log files, such as:
  - EAServer log
  - HTTP error log
  - HTTP request log
  - HTTP servlet log

See “Logging” on page 4 for more information on log files.

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 `setenv.sh` (UNIX) or `setenv.bat` (Windows):
  - PATH
  - JAGUAR
  - SQLANY
  - CLASSPATH
  - BOOTCLASSPATH
  - LD_LIBRARY_PATH
  - JAGUAR_JDK12, JAGUAR_JDK13, and JAGUAR_JDK14

Not all the 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 `user_setenv.sh` (UNIX) or `user_setenv.bat` (Windows) for additional settings.

See “Configuration issues” on page 35 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

A number of log files provide useful troubleshooting information and are
described in this section.

EAServer log

This log file contains the server version, build number, listener addresses, Java
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 traces”
on page 10.
- Call System.out.print from a Java component.
- Call jaguar.server.Jaguar.writeLog method in a Java component. See the
- Call JgServer::WriteLog from an ActiveX component. See the EAServer
  API Reference Manual for an example.
- Call JagLog from a C or C++ component. See “Handle errors in your C
  component” in Appendix C, “Creating C Components,” in the EAServer
- Utilize the ErrorLogging object for a PowerBuilder component. See “Error
  logging service” in the PowerBuilder Application Techniques manual for
  usage information.

The preconfigured server writes messages to the Jaguar.log file. Change the
log file name for each new server that you create.
If you have changed the default configuration, or run user-defined servers, the log file may not be Jaguar.log. To change or verify the log file name:

1. In EAServer Manager, highlight the server, right-click and select Server Properties.
2. On the Log/Trace tab, enter the Log File Name and click OK.

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**

Beginning with version 5.0, EAServer includes a configurable logging mechanism that allows integration with the JDK 1.4 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 also configure different log profiles for the debug and production server versions.

You can use the following logging subsystems:

- The built-in EAS subsystem, which offers the same functionality available in EAServer 4.x versions, plus several enhancements:
  - The ability to configure log levels so that messages below a specified level of severity are discarded
  - Support for different logging configurations in the debug and production servers
  - Optional archiving and compression of previous log file versions
  - More control over message formatting

- Apache Log4j, which is commonly used on large projects. For more information, see the Apache Log4j Documentation at http://jakarta.apache.org/log4j/docs/api/overview-summary.html.

- The Java Logging package, included in JDK 1.4. This API is Sun’s 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.
Logging

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
- `IJagServer.writeLog` from ActiveX 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 log profiles


Server diagnostics

You can export a diagnostic log file that describes the server version and environment in detail. To use this feature, highlight the server icon, choose Export Diagnostic Log, and enter a file name.

HTTP logs

The HTTP error logs contain information about client requests. The default location for the log files is the EAServer *bin* subdirectory. You can change the name and location of any of the HTTP logs on the HTTP Config tab of the Server Properties dialog box. You can also enable or disable request logging on this tab.
CHAPTER 1 Monitoring Techniques

HTTP error log The HTTP error log contains information about failed client request errors, such as requests for nonexistent files. The default file name is `<server_name>httperror.log`.

HTTP request log The HTTP request log tracks all requests to the HTTP listener. The default file name is `<server_name>httprequest.log`.

HTTP servlet log The HTTP servlet log contains information logged by the EAServer servlet engine; this is in addition to the servlet-related messages recorded in the server log file. The default file name is `<server_name>httpservlet.log`.

IIOP log

You can use the IIOP log property of EAServer to trace IIOP session activity by setting `com.sybase.jaguar.server.iiop.log` to `true` and restarting the server. All IIOP traffic between all clients and the server is written to the server log.

The log may be helpful in diagnosing these problems:

- 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 diminish server performance. Use IIOP logging only for short durations; turn it off as soon as you gather the information you need.

C/C++ and PowerBuilder clients For C/C++ and PowerBuilder clients, Table 1-13 on page 14 describes how to direct the ORB to log IIOP protocol trace information.
Using stack traces

Installation log

Beginning with EAServer 5.0, you can run the installer in debug mode, and direct the output to a log file using the following command:

```
JRE_1_4\bin\java -Dis.debug=true -Deas.debug=true -jar EAS500.jar > log_file
```

where `log_file` is the name of the file where you want to save the output.

Other useful data

Other useful data comes from:

- The `<ServerName>_boot.log` file is generated as the server is starting. If the server fails to start, this file may help you find the reason for the failure.
- Server console messages, which may provide additional error information.
- The file `Repository/Server/<ServerName>_admin`. If the server starts in Admin mode, this file contains an explanation.
- Licensing output in the log file.
- Client-side log files. See “EAServer traces” on page 10 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. It is better to 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 76.

**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 Windows**

1. Change to the `%JAGUAR%/bin` directory.
2. Open the `serverstart.bat` file, and look for the line that contains:

   ```
   jagsrvagent.exe -servername %ServerName% .... 2>&1 tee ...
   ```
3. Remove “2>&1 tee”, then save and close the file.
4. Restart EAServer.
5. Ctrl+Break

**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 78.
Using stack traces

EAServer traces

EAServer provides trace facilities to help you obtain information on different areas of activity:

- Server level—see Table 1-1 on page 10.
- Open Server—see Table 1-2 on page 11.
- Methods as Stored Procedures (MASP)—see Table 1-3 on page 11.
- Listeners—see Table 1-4 on page 11.
- Security—see Table 1-5 on page 12.
- Servlets and Web applications—see Table 1-6 on page 12.
- Components—see Table 1-7 on page 12.
- Message service—see Table 1-8 on page 12.
- Java class loading and unloading—see Table 1-9 on page 13.
- Additional Java and JVM tracing—see Table 1-10 on page 13.
- Connections and transactions—see Table 1-11 on page 13.
- Web server redirector—see Table 1-12 on page 14.
- PowerBuilder or C/C++ clients—see Table 1-13 on page 14.
- Other related properties—see Table 1-14 on page 15.

The tables below describe the options available in each area.

Note  Some properties exist only in certain EAServer versions. Check the documentation and release notes for your EAServer version.

For information about setting the com.sybase.jaguar.* properties, see Appendix B, “Repository Properties Reference,” in the EAServer System Administration Guide. Consult PowerBuilder Application Techniques for information about how to configure the PowerBuilder properties.

Table 1-1: Server-level tracing

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.server.lwc.debug</td>
<td>Logs the EJB-to-EJB calls that use the lightweight container.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.logspid</td>
<td>Includes the thread identifier in each message in the log file.</td>
</tr>
<tr>
<td>com.sybase.jaguar.threadmonitor.trace</td>
<td>Logs peak number of active and waiting threads in server log. If com.sybase.jaguar.threadmonitor.callstats is nonzero, also enables printing call statistics.</td>
</tr>
</tbody>
</table>
### Table 1-2: Open Server tracing

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.server.tracenetdriver</td>
<td>Traces network driver requests</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.tracenetrequests</td>
<td>Traces requests to transport control layer (an internal EAServer library that serves as an interface to network drivers)</td>
</tr>
</tbody>
</table>

### Table 1-3: MASP tracing

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.server.traceattentions</td>
<td>Tracks TDS (Tabular Data Stream™) protocol attentions</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.tracetdsdata</td>
<td>Tracks TDS packet contents</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.tracetdshdr</td>
<td>Tracks TDS header contents</td>
</tr>
</tbody>
</table>

### Table 1-4: Listener tracing

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.server.http.cache.debug</td>
<td>Writes static page cache information to the server log.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.iop.log</td>
<td>IIOP logging. Output is verbose.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.iop.log.ac</td>
<td>IIOP logging for events that occur after the server begins accepting messages. Less output than com.sybase.jaguar.server.iop.log; omits server start-up messages.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.http.elfenable</td>
<td>Determines if the extended log file format is used to write information to the request log.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.http.elfitems</td>
<td>Determines the items that are logged and the order in which they are written to the request log. Fourteen items can be logged, including request, status, bytes, and cookie—see Appendix B in the EAServer System Administration Guide for the complete list.</td>
</tr>
</tbody>
</table>
Using stack traces

**Table 1-5: SSL connection tracing**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.security.logpeerIP</td>
<td>If this property is set to true in an SSL listener's security profile, a client's IP address is logged when an SSL connection cannot be established. The default value is false.</td>
</tr>
</tbody>
</table>

**Table 1-6: Servlets/Web applications tracing**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.server.servlet.trace</td>
<td>Sets trace logging in EAServer servlet execution engine</td>
</tr>
<tr>
<td>com.sybase.jaguar.webapplication.sectrace</td>
<td>Sets security implementation tracing</td>
</tr>
</tbody>
</table>

**Table 1-7: Component tracing**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.component.debug</td>
<td>Logs trace information for instance lifecycle events (creation, destruction, pooling and so on)</td>
</tr>
<tr>
<td>com.sybase.jaguar.component.trace</td>
<td>Logs methods and parameters</td>
</tr>
<tr>
<td>com.sybase.datawindow.trace</td>
<td>Set to true to enable tracing for the Web DataWindow component (DataWindow/HTMLGenerator)</td>
</tr>
</tbody>
</table>

**Table 1-8: Message service tracing**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jms.debug</td>
<td>Enables message service JMS debugging</td>
</tr>
<tr>
<td></td>
<td>To help debug your JMS client application, you can enable tracing by setting this property to true in the InitialContext object. When you enable tracing, diagnostic messages are printed in the console window. By default, tracing is disabled. This code sample illustrates how to set the tracing property: Properties prop = new Properties(); prop.put(&quot;com.sybase.jms.debug&quot;, &quot;true&quot;); javax.naming.Context ctx = new javax.naming.InitialContext(prop);</td>
</tr>
<tr>
<td>cms.debug</td>
<td>Enables all message-service debugging</td>
</tr>
<tr>
<td>Note</td>
<td>You can set this property and the following two properties in the MessageServiceConfig.props file, located in $JAGUAR/Repository/Component/CtsComponents.</td>
</tr>
<tr>
<td>cms.debug.network</td>
<td>Enables message service network-level debugging</td>
</tr>
<tr>
<td>cms.debug.session</td>
<td>Enables message service session-level debugging</td>
</tr>
</tbody>
</table>
### Table 1-9: Java class loading/unloading tracing

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.server.jvm.verbose</td>
<td>Logs all Java classes loaded and the location where each class file was read</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.classloader.debug</td>
<td>Logs information about loading classes from custom class lists defined for the component, package, application, or server</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.jvm.verboseGC</td>
<td>Logs information when memory is freed by the Java garbage collector</td>
</tr>
</tbody>
</table>

### Table 1-10: Additional Java tracing

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.server.jcm.trace</td>
<td>Traces Java Connection Manager (JCM) activity.</td>
</tr>
<tr>
<td>cmp_driver_debug</td>
<td>Tracks what the driver wrapper is doing; for CMP (container-managed persistence) wrapper driver.</td>
</tr>
<tr>
<td>cmp_driver.print_warnings</td>
<td>Logs all database warning messages received by driver wrapper; for CMP wrapper driver.</td>
</tr>
<tr>
<td>cmp_driver.trace_commit</td>
<td>Traces transaction commit, rollback, and autoCommit changes for CMP wrapper driver.</td>
</tr>
</tbody>
</table>

For more information about the cmp_driver properties defined in Table 1-11, see Chapter 4, “EJB CMP Tuning,” in the EAServer Performance and Tuning Guide.

### Table 1-11: Connection and transaction tracing

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sybase.jaguar.server.jvm.debugging</td>
<td>Enables Java in-server predebugging in the pre-4.1.3 debug server. In EAServer version 4.1.3 and later, simply use the debug server.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.jvm.debug.options</td>
<td>Enables additional JVM options to configure in-server Java debugging.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.jvm.displayOptions</td>
<td>Writes the current JVM properties to the server log.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.jvm.nojit</td>
<td>When using the Client Hotspot JVM, indicates whether the JIT (just-in-time) compiler is disabled. In EAServer versions earlier than 4.1, the default for new servers is true. In 4.1 and later, the default is false.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.jvm.options</td>
<td>Specifies options to pass to the server’s Java virtual machine.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.jvm.minHeapSize</td>
<td>Specifies the minimum heap size for the server’s Java virtual machine.</td>
</tr>
<tr>
<td>com.sybase.jaguar.server.jvm.maxHeapSize</td>
<td>Specifies the maximum heap size for the server’s Java virtual machine.</td>
</tr>
</tbody>
</table>
**Using stack traces**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cmp_driver.trace_connect</code></td>
<td>Traces connect and reconnect operation for CMP wrapper driver.</td>
</tr>
<tr>
<td><code>cmp_driver.trace_create</code></td>
<td>When using the Sybase (CMP JDBC wrapper for Sybase jConnect driver), traces the creation of semipermanent stored procedures.</td>
</tr>
<tr>
<td><code>cmp_driver.trace_execute</code></td>
<td>Traces the execution of stored procedures and SQL command batches; for CMP wrapper driver.</td>
</tr>
<tr>
<td><code>cmp_driver.trace_execute_ms</code></td>
<td>Traces the execution of stored procedures and SQL command batches that take longer than the specified number of milliseconds. Takes precedence over <code>cmp_driver.trace_execute</code>; for CMP wrapper driver.</td>
</tr>
<tr>
<td><code>com.sybase.jaguar.conncache.cmp_stats</code></td>
<td>Seconds between “Table Statistics” entries in the server log when using CMP.</td>
</tr>
<tr>
<td><code>com.sybase.jaguar.server.jta.trace</code></td>
<td>Java Transaction API; set to an integer for the desired trace level.</td>
</tr>
</tbody>
</table>

For information about setting the Web server redirector properties, see the *EAServer Installation Guide* for your platform.

**Table 1-12: Web server redirector tracing**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`Connector.LogLevel = [inform</td>
<td>error</td>
</tr>
<tr>
<td><code>Connector.SessionId ID</code></td>
<td>Traces requests from connectors. Set this directive in the redirector config file so the connector appends the value of <code>Connector.SessionId</code> to the URL forwarded to EAServer. EAServer writes the URL to the server’s HTTP request log to help in debugging.</td>
</tr>
</tbody>
</table>

**Table 1-13: PowerBuilder or C/C++ clients**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ORBLogIIOP</code></td>
<td>Specifies whether the ORB should log IIOP protocol trace information. If set to true, enables logging. The default is false. This parameter can also be set in the <code>JAG_LOGIIOP</code> environment variable. When this parameter is enabled, you must set the <code>ORBLogFile</code> option (or the corresponding environment variable) to specify the file where protocol log information is written.</td>
</tr>
</tbody>
</table>
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:

```java
try {
```
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 `NamingException`, other errors can be embedded, such as the `NamingException` root cause. Often, you must retrieve details about the embedded error to 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, you should 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 down 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:

- EAServer Manager monitoring – shows IIOP current and peak sessions, requests, and bytes read.
- IIOP output logging – displays IIOP session activity. See “Logging” on page 4 for more information.

HTTP

EAServer Manager monitoring displays the following HTTP information:

- Current and peak sessions
- Requests
- Bytes read
Tracing network problems

- Logging of HTTP requests, HTTP statistics, HTTP errors, and servlet requests. See “Logging” on page 4 for more information.

TDS

You can monitor Tabular Data Stream (TDS) traffic using:

- EAServer properties to log output; see Table 1-3 on page 11.
- The jConnect Ribo utility to capture the TDS stream and translate it to text.

CHAPTER 2

Common Problem Areas

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation issues</td>
<td>19</td>
</tr>
<tr>
<td>Server crashes, hangs, or disappears</td>
<td>20</td>
</tr>
<tr>
<td>Server slows or runs out of memory</td>
<td>27</td>
</tr>
<tr>
<td>Server starts in Admin mode</td>
<td>27</td>
</tr>
<tr>
<td>Connection problems</td>
<td>28</td>
</tr>
<tr>
<td>Application issues</td>
<td>28</td>
</tr>
<tr>
<td>Web server redirector plug-in issues</td>
<td>33</td>
</tr>
<tr>
<td>Configuration issues</td>
<td>35</td>
</tr>
<tr>
<td>System-level issues</td>
<td>38</td>
</tr>
</tbody>
</table>

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*:

2. Click Application Servers.
3. Click EAServer.
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 to successfully isolate the cause.

Server crashes

A server failure (crash) 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. Set the JAGUAR_RANDOMSEED variable, which is known to resolve start-up crashes in some instances, especially if the server takes a long time to start. See Chapter 3, “Creating and Configuring Servers,” in the EAServer System Administration Guide for details.

2. Make sure that the PATH and CLASSPATH variables are correctly set. Also verify which DLL or library versions were loaded, and class load locations. For Windows, see “ListDLLs” on page 71. For UNIX, use `pmap` or another command or tool appropriate for your platform.

3. If operating in a cluster environment:
   - Check cluster start-up/check settings
• Ensure all members have the same listener names and types

4 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 are:
   a Load licensing files, check options
   b Load network libraries for listeners
   c Initialize JVM
   d Initialize various data structures
   e Load server ClassLoader
   f Load JCM (Java Connection Management) caches, connectors, and resources
   g Start services:
      • Repository, GC, JCM, and Naming ServletService
      • User services; for example, the message service

Up to this point, the server is in Admin mode. If there are no problems, the server begins accepting regular client connections, as indicated by the console message “Accepting connections.” If a problem occurs, the server may remain in Admin mode as described in “Server starts in Admin mode” on page 27.

5 Use the JagRepair server to:
   • Check whether the server is looking for listeners that do not exist
   • Temporarily prevent user-defined services from starting automatically
   • Check for connection caches configured without a DLL or library name, or other inappropriate settings

6 Set com.sybase.jaguar.server.jvm.verbose to true.
   This shows all JVM initialization details, listing each JAR file that is opened and each class file that is loaded.
   Try a different JVM version. If the problem persists, continue with the steps below.

7 In the server start batch file, set ECHO to on. This provides more output and may help determine the point of failure.
Gather crash information such as the load address. For Windows, see “ListDLLs” on page 71; for UNIX, use `pmap` or another command or tool appropriate for your platform.

When the server crashes, the first step is to identify the crash address. You can obtain this from the console or log file. For example:

```
jagsrv.exe Exception access violation (0xc0000005)
   Address: 0x02af57c6
```

You can 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.

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.

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, take the following steps:

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

2. Use the Runtime Monitoring feature in EAServer Manager to find out which components are active, the connections being used, and so on—see “Runtime monitoring tools” on page 69.

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

4. Explore what is happening in the application or component. Look for common application trouble spots—see “Generic issues” on page 28.

5. Check the component and server property settings.

6. Debug the component.

7. Check the server configuration, specifically:
   - Patch levels—see “EAServer log” on page 4 for details.
CHAPTER 2    Common Problem Areas

- Resource and memory usage.
- DLL versions, library versions, and Java class loading—see “Verifying your configuration” on page 36 for library and class information, and “ListDLLs” on page 71.

8 Invoke a service component just before the crash to write information to the EAServer log—see “ListDLLs” on page 71:
   - JVM properties
   - Free and total memory
   - EAServer monitoring data
   - EAServer memory dump

9 If a C++ component is involved, try running it within a dedicated external process, identified by the com.sybase.jaguar.component.cpp.process property.

   There are restrictions on running externally, and not all C++ components are suitable candidates. See “Running C++ components externally” in Chapter 14, “Creating CORBA C++ Components,” of the EAServer Programmer’s Guide for details.

10 Run the debug server for more information in the dump file. Use the stack trace, dump file, and crash address for advanced troubleshooting—see “Stack traces, dump files, and core files” on page 76.

Server crashes intermittently

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

- Ensure that sanity checks are enforced on the connection cache; 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 awhile, then stops responding to requests:

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:
   - Connect to the IIOP listener port using EAServer Manager. 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 TDS listener port by executing a MASP call. Use isql for an Adaptive Server Anywhere or Adaptive Server Enterprise database. Use a vendor-supplied tool for other database types.
     For example, use the following isql command to return the server build number, using the Management component:
     ```
     isql -Ujagadmin -P -SJaguar
     1> exec Jaguar.Management.getServerBuildNumber
     2> go
     ```
     See Appendix A of the EAServer Programmer’s Guide for more information about MASP.
   - Check the HTTP listener port at http://machine_name:8080 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, and include additional items `com.sybase.jaguar.server.http.elfenable` (show extended items) `com.sybase.jaguar.server.http.elfitems` (show item information).
     - 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 connection cache:
• Ping the connection cache in EAServer Manager. If ping fails, use isql (or a vendor-supplied tool for a non-Sybase database) to try a simple SQL request outside EAServer.

Using a database monitoring tool, investigate what is happening on the database server side. Check locks held.

• Look at the cache in EAServer Monitor. Verify that you have not reached the maximum number of connections for the cache. Check the number of connections: active, pooled, forced, peak, and so on.

• If you were connecting 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 TDS or SQL requests passed. For information about using Ribo, see the JDBC for jConnect Installation Guide at http://sybooks.sybase.com/jc.html.

• 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?

• Set com.sybase.jaguar.component.debug and com.sybase.jaguar.component.trace to on.

• 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 with Apache’s 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:

Troubleshooting Guide
Server crashes, hangs, or disappears

- Verify whether EAServer maximums are being reached; specifically, check if client sessions (IIOP + TDS) are at their maximum, or if the maximum thread limit (IIOP + HTTP) is reached.

**Note** In versions of PowerBuilder earlier than 8.0.3, instance data stores use twice as many threads as they currently do.

- 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 *jagsrv* 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” on page 4 for details.
2. On UNIX, check to see if a core file was generated. On Windows, check for a `.log` or `.dbg` file.
3. Check the *jagsrv* 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:
     ```bash
     serverstart.sh > err.out 2>&1
     ```
     Windows:
     ```bash
     serverstart.bat >info.out 2>err.out
     ```
   - On Windows, set the crash handler to *userdmp.exe* instead of Dr. Watson. See Chapter 6, “Advanced Topics,” for more on debugging tools.
   - Try the debug server.
5. Implement a service component to periodically write information to the server log.
Server slows or runs out of memory

If the server slows down or runs out of memory, you can investigate several areas:

1. In EAServer Manager, select Servers | <Server Name> | Runtime Monitoring, and monitor the system to check pooling, peaks, maximums, sessions, and so on.
2. Check system sizing.
3. 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 69.
4. Review EAServer timeout properties. If no timeout is set, this leads to indefinite wait for transactions, components, or methods (often the EAServer default).
5. Look for application trouble spots—see “Application issues” on page 28.
6. Check for memory leaks. For Java profiling, use OptimizeIt. On Windows, use the Performance Monitor.
7. Implement a service component to periodically write information to the server log.

Server starts in Admin mode

In some cases, EAServer may start in Admin mode. In this mode, the server accepts only IIOP client connections from EAServer Manager or clients that are coded to connect in Admin mode. Normal client connections are refused.

The server may start in Admin mode for a variety of reasons, for example:

- The server may belong to a cluster, but the server configuration is not synchronized with the cluster configuration.
- The message service is configured, but cannot start.

To find out why the server is in Admin mode, check the server log or the contents of the Repository/Server/<Servername>.props file, where Servername is the name of the server.
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 12, “Creating Java CORBA Clients,” in the EAServer Programmer’s 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

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 EAServer Thread Manager to allow EAServer to manage threads that must be spawned by a component.

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 at the session level only.

**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, you can use the Verifier tool to verify that it is valid. For details, search for “Verifier Tool” on the Java Web site at [http://java.sun.com](http://java.sun.com).

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.

**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 become out of sync. You must generate a new skeleton class if you do any of the following:
Application issues

- 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 unless they are run as external processes. If you are having trouble with C++ components, Sybase suggests that you configure the components to run as external processes. If this is not possible, try moving them to another server and repository, so that if they crash, other components are not impacted. See “Running C++ components externally” in Chapter 14, “Creating CORBA C++ Components,” in the EAServer Programmer’s Guide for details.

PowerBuilder component issues


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 54.

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 instead. See TechNote 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, refer to the PowerBuilder section of the EAServer Performance and Tuning Guide. You can find the online version of the PowerBuilder section at http://www.sybase.com/detail?id=1019504#pb.

**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 an erroneous 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 the Guidelines for Code Set Interoperability with PowerBuilder and EAServer at http://content.sybase.com/detail?id=1028793.

**Trace flags** All EAServer trace flags are described in “EAServer traces” on page 10.

**Client log file** To configure a client log file, see Table 1-13 on page 14.
Application issues

Error handling  See “PowerBuilder error handling” on page 54.

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 goes away, the object persists until a timeout in the server. 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 since their storage persists for the lifetime of the component instance, even if that instance is not currently active.
- 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 voting on the transaction to ensure correct lifecycle execution of components.
- Set blobs to null when finished.
Other design issues


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.

All Web server redirectors

If you are using one of the Web server redirector plug-ins, you must set the value of the com.sybase.jaguar.listener.http.connector_events property to true; otherwise, the HTTP responses that are sent to clients may be incorrect. The default value of the com.sybase.jaguar.listener.http.connector_events property is false.

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 dll directory, or if you are using the debug version of the server, verify that the files are in the dll/debug directory:
   • lijctsseect.dll
   • libjcc.dll
   • libjsybscl.dll
   • libjintl.dll
   • libjeas_iis.dll
2 Verify that the PATH system variable includes the EAServer dll directory, or the dll/debug directory for the debug server.

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:\winn\system32\inetpub, the value of WSPLUGIN_CONFIG_FILE should be c:\winn\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 8080:


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

   Connector.IIS.LogLevel inform

   the log file should display the following:

   Wed May 21 16:53:58 2003 INFO: ws: 1de1090, URL: [http://prodhost:8080], protocol: [http], host: [prodhost], port: [8080] 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.
10 If the icon for your virtual directory looks like the IISADMIN icon:
   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.
Configuration issues

Verifying your configuration

All platforms

- Verify that you can ping your connection cache.

- Check the deployment properties of the application’s components. See Chapter 14, “Importing and Exporting Application Components,” in the EAServer System Administration Guide for information about deploying components.

- Look for Java classes loading from unexpected locations. Class loader logging is enabled by com.sybase.jaguar.server.jvm.verbose.

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

- Verify that maximums and limits are set appropriately for the production environment:
  - EAServer properties (caches, sessions, threads, and so on)
  - JVM

- Ensure that the listener ports are not already in use and that no two EAServer listeners are defined to use the same port number.

- Check that 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.

Note  EAServer includes libjcc.dll in the EAServer dll directory. PowerBuilder may include the file as well, in the %SYBASE_SHARED%/PowerBuilder directory, since this file is needed on PowerBuilder client installations. Sybase recommends that you not copy the libjcc.dll 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 EAServer Manager 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 serverstart script, the problem is less likely to occur; users must modify either the script or environment to cause the problem.

- If you specify the CacheName in the dbparm property of the transaction object, make sure you check the Enable Cache-By-Name access box.
Solaris platforms

Verify that LD_LIBRARY_PATH includes:

```
$JAGUAR/lib:$SQLANY/lib:$JAGUAR/intersolv/lib
$JAGUAR/jdk/jdk_latest/lib/sparc/libthread/5.5.1
$JAGUAR/jdk/jdk_latest/lib/sparc/native_threads
$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/bin/x11/)
```

Verify that the THREADS_FLAG variable is either not set or is set to “native_threads.”

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

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

The %JAGUAR%\bin\serverstart.bat and %JAGUAR%\html\classes\jagmanager.bat files modify the PATH and CLASSPATH variables. Look at these files to investigate problems with either of these variables.

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. Resource allocation problems associated with running the server as a service may not occur when running the server from the console.

If you run EAServer as a service, all the ODBC data sources for defined connection caches must be system data sources. The server cannot find user data source names.
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 TechNote 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 TechNote 1010600, Binding a PowerBuilder Process to a CPU on Windows NT or 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 an 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 if the EAServer process is reaching the virtual bytes limit, see “Evaluating Windows memory” on page 69.

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://www.microsoft.com.
System-level issues
CHAPTER 3  
Performance Issues

Resources

A number of documents are available in MySybase to help you improve performance for EAServer applications. They include:

- **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++, ActiveX, and PowerBuilder components
  - Message service
  - Client applications
  - Database access
  - Web Services Toolkit
  - Runtime monitoring

- **What's New in EAServer**
  Chapter 1, “New Features in EAServer Versions 5.0, 4.2, 4.1.x, and 4.0,” surveys key performance tuning features.

- **EAServer System Administration Guide**
  Chapter 7, “Load Balancing, Failover, and Component Availability,” explains how to optimize performance for your EAServer cluster by adjusting the load across the servers.
Chapter 12, “Runtime Monitoring,” describes how you can use the File Viewer, Runtime Monitor, and OTS Transaction Monitor to track EAServer’s performance.

- *EAServer Programmer’s Guide*

  Chapter 7, “Advanced Features,” 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:

```java
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 11, “Creating CORBA Java Components” of the EAServer Programmer’s 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 to the server’s 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 in Java

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.lookup` 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:

```java
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_ROLLED_BACK 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_PERMISSION 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.
   ...

The example does not show all of the possible system exceptions. See the CORBA/IIOP 2.3 specification for a list of all the possible exceptions.

### User-defined exceptions in Java

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:

```idl
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;

**Note** As per the CORBA specification, methods defined on exceptions cannot be marshalled to the client; that is, information provided with an exception must be in a public variable.

---

**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; must be defined in the component’s IDL definition

**CORBA system exceptions in C++**

The CORBA specification defines the list of 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:

```c++
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_PERMISSION& 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 5, “Defining Component Interfaces” of the EAServer Programmer’s Guide.

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

Error handling in ActiveX components

Handle errors that may occur during a method call as follows:

- Call the JagServer:WriteLog method to write a description of the error to the log file.

JagAxWrap.dll must be registered on your machine. If you are developing on a machine that already has EAServer installed on it, JagAxWrap.dll is already registered.
You can also generate an ActiveX automation exception with text that describes the error. EAServer returns the text of the exception to the client. Java clients receive the message as a Java exception (class com.sybase.jaguar.util.JException) and ActiveX clients receive the message as an ActiveX automation exception.

In general, if an error prevents completion of a desired task (such as database updates that represent a new sales order), you should generate an ActiveX automation exception to send a concise description of the problem to the client. Messages sent to the client should be concise and contain language suitable for display to the end user. You can record more detailed messages in the log file.

IDL user-defined exceptions are not supported.

Never write your component to send error messages to the console to display dialog boxes. Servers run unattended; showing a dialog box does nothing but hang the thread that executes your component.

Handling exceptions in ActiveX clients

Always make sure that your application handles exceptions gracefully. At minimum, you should display the exception text, which may aid debugging.

Errors in ActiveX proxy execution can be handled as ActiveX exceptions, or inline using a try/catch model similar to the structured exception-handling model in the C++ and Java languages.

Using an ActiveX error handler

By default, the ActiveX proxy raises an ActiveX exception when an EAServer component method raises an exception or an internal error occurs. Visual Basic and most other ActiveX scripting tools do not allow you to handle these errors inline. Instead, control transfers to an error handler (specified by an error goto in Visual Basic) or to a system-wide error dialog box. To handle proxy errors inline, you must enable inline exception handling as described in “Handling exceptions inline” in the EAServer Programmer’s Guide.
Structure of an ActiveX exception

In C++, the OLE EXCEPINFO structure describes an ActiveX exception. Different ActiveX-enabled IDEs provide different mechanisms for applications to obtain the EXCEPINFO structure contents.

In Visual Basic, exceptions are mapped to the built-in Err object. The exception number maps to Err.Number and the description is available as Err.Description. You can handle exceptions by activating error handling code with the **On Error Goto** statement or by checking whether Err.Number is > 0.

IDL user-defined exceptions are not supported and are mapped to error number 9000.

Using error pages

You can create error pages to customize error and exception reports that are sent to clients. 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 declare error pages at the server level and for a Web application or JavaServer Page (JSP).

You can specify HTML and JSP files to send in response to HTTP error codes and to Java exceptions thrown in JSPs or servlets. For information about JSP error pages, see “JSPs as error pages” on page 52.

Server-level error pages

To set up server-level error pages, set the value of the `com.sybase.jaguar.server.servlet.error-page` property on the Advanced tab of the Server Properties dialog box. Use the following syntax to define a comma-delimited list of complex properties. Each property includes a location, and either an error code or an exception field.

```properties
(location=/file.jsp,error-code=code),
(location=/exception.htm,exception=java.lang.Exception)
```

where:

- The locations of error pages `file.jsp` and `exception.htm` are relative to the HTML document root.
- `code` is the error code that triggers the error page.
CHAPTER 4 Exception Handling

- `java.lang.Exception` is the fully-qualified Java class name of the exception that triggers the error page.

Error pages for Web applications

When an exception is thrown, the servlet engine searches the error page mappings for the exception and its super classes. For example, assume `AException` extends `BException` and `BException` extends `CException` and `CException` extends `java.lang.Exception`. When `AException` is thrown, EAServer checks if `AException` is mapped. If not, EAServer checks if `BException` is mapped, and so forth.

Adding an error page

Use the following steps to add an error page:

1. Display the File Refs tab in the Web Application Properties dialog box.
2. Under Error Mapping, click Add. A new row is added to the mapping table with default settings.
3. Place the cursor in the Error/Exception cell, and enter the error number or Java exception class name.
4. Place the cursor in the URL cell, and type the path to the file, relative to the Web application’s context root; for example, `/etc/error404.html`.
5. Verify that the file exists in your EAServer installation directory and can be read by the server process. For example, the path `/etc/error404.html` corresponds to the `Repository/WebApplication/web_app/etc/error404.html` file in your EAServer installation, where `web_app` is the name of the Web application.

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.
Using error pages

Uncaught exceptions

You can handle any exceptions that are thrown from the body of the implementation class and are not caught by using an error page that you specify using a page directive. 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:

```jsp
<%@ 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.

Saving Java source code

Normally, EAServer deletes the Java source code after compiling a JSP. To keep the generated source code to view or use in a debugger:

1. Display the properties for the Web application in which the JSP is installed.
2. On the Advanced tab, set the com.sybase.jaguar.webapplication.keepgenerated property to true.

JSPs as error pages

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

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

```jsp
<%@ 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.
CHAPTER 4 Exception Handling

<% 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 of the Application Techniques manual
- “Testing and debugging the component” in Chapter 22 of the Application Techniques manual
- “Handling errors” in Chapter 23 of the Application Techniques manual
- “Troubleshooting connections” in Chapter 23 of the Application Techniques manual
- “Handling Data Windows Errors” in Chapter 2 of the DataWindow Programmer’s Guide

Unhandled PowerBuilder exceptions

If an unhandled fatal exception is raised by a PowerBuilder component running in EAServer, it 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. Rather than allow EAServer to continue running in an unstable state, you may want to restart or shut down the server.

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Resulting behavior when a fatal error occurs in a PowerBuilder component</th>
</tr>
</thead>
<tbody>
<tr>
<td>continue</td>
<td>EAServer continues running, and a CORBA_TRANSACTION_ROLLEDBACK exception is thrown.</td>
</tr>
<tr>
<td>restart</td>
<td>EAServer restarts automatically.</td>
</tr>
<tr>
<td>shutdown</td>
<td>EAServer shuts down.</td>
</tr>
</tbody>
</table>

If the PBOnFatalError variable is not set, the default value is “continue.”
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:

<table>
<thead>
<tr>
<th>Source</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRVLIB</td>
<td>Generated by EAServer</td>
</tr>
<tr>
<td>DEBUG</td>
<td>Generated by a component when the debug property is enabled</td>
</tr>
<tr>
<td>TRACE</td>
<td>Generated by a component when the trace property is enabled</td>
</tr>
<tr>
<td>CORBA</td>
<td>Generated by CORBA client ORB</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Message text (or text fragment)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannot find interfaces file</td>
<td>See “Connection problems” on page 28 and “Configuration issues” on page 35.</td>
</tr>
<tr>
<td>Cannot find localization files</td>
<td>See “Connection problems” on page 28 and “Configuration issues” on page 35.</td>
</tr>
<tr>
<td>Cannot start network listener</td>
<td>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. See “Configuration issues” on page 35 for information about resolving start-up problems.</td>
</tr>
</tbody>
</table>
| ClassCastException error        | If you see NamingContext exceptions with root-cause exception ClassCastException when calling javax.naming.InitialContext.lookup, check for the following problems:  
• You may be casting to an incorrect type (check the class name of the object returned by lookup).  
• Your component has refresh enabled, and the custom class list does not contain some required classes.  
• 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 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:  
    sp_procxmode "sp_myproc", "anymode" go  
    You can run the stored procedure sp_myproc only in unchained transaction mode. The SET CHAINED OFF command causes the current session to use unchained mode. |
<table>
<thead>
<tr>
<th>Message text (or text fragment)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORBA marshall exception</td>
<td>You may have tried to return a NULL object, which CORBA forbids. 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.</td>
</tr>
</tbody>
</table>
| CORBA:: NO_PERMISSION         | The user is not authorized to perform the requested operation. For example, the SessionLOOKUP call throws a NO_PERMISSION exception when you request a component instance and the user lacks permission to instantiate that component. Check permissions for the component, package, and user. 
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. 
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.) 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. |
### Error messages

<table>
<thead>
<tr>
<th>Message text (or text fragment)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORBA::OBJECT_NOT_EXIST</td>
<td>This message can mean that the package or component is not installed on the server. 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.</td>
</tr>
<tr>
<td>CORBA::TRANSIENT</td>
<td>For EJB clients, this exception is the root cause of the java.rmi.RemoteException thrown by the EJB stub. 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. Use one of the following corrective actions: • Set the property com.sybase.jaguar.component.debug=true to get more information; or • Use a timestamp column in your database table; or • Disable optimistic updates as appropriate for your application. See Chapter 25, “Managing Persistent Component State,” in the EAServer Programmer’s Guide for more information on concurrency control.</td>
</tr>
<tr>
<td>Could not start thread</td>
<td>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.</td>
</tr>
<tr>
<td>CtsComponents::CreateException</td>
<td>See javax.ejb.CreateException error.</td>
</tr>
<tr>
<td>CtsComponents::FinderException</td>
<td>See javax.ejb.FinderException error.</td>
</tr>
<tr>
<td>Message text (or text fragment)</td>
<td>Explanation</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 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. In PowerBuilder component code, you typically have a series of lines that 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, and the definition must come before this statement:  
  
```sql
CONNECT USING SQLCA;
```

PowerBuilder uses the DBMS string to build the DLL or library name that it needs. 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. Connect to EAServer Manager to confirm that EAServer is running. 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. Check the EAServer user ID and password specified on the Server tab in the Component Generator properties. 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:  
  
- Standard datatypes  
- Structures  
- Custom class user objects that have been deployed as EAServer components  
- ResultSets  

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. |
### Error messages

<table>
<thead>
<tr>
<th>Message text (or text fragment)</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| The Instruction at *** referenced memory at ***, the memory could not be written              | You may see this message if the JAGUAR_RANDOMSEED variable is not set, or is set to a value that does not match the name of an accessible file.  
See Chapter 3, “Creating and Configuring Servers,” in the EAServer System Administration Guide for more information. If that does not resolve the problem, see “Server crashes, hangs, or disappears” on page 20. |
<p>| 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. Finder methods throw javax.ejb.FinderException if no rows match the specified search criteria.                                                                                             |
| java.lang.NoClassDefFoundError                                                                | A class definition could not be found. Make sure the class file is in the CLASSPATH. 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 set the property back to false when you are finished. |</p>
<table>
<thead>
<tr>
<th>Message text (or text fragment)</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| 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:  
  • The server address specified with the Context.PROVIDER_URL property is incorrect or the server is not running.  
  • Authentication with the specified credentials failed.  
  • 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.  
  • Check the EJB references in EA Server Manager to be sure the values are correct.  
  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 EA Server message service topic or queue, without using the same connection cache. The component must use the same connection cache as the EA Server message service. |
| jmsException | An exception has occurred with the message service. The ExceptionListener provides access to the error information—see Chapter 29, “Creating connections,” in the EA Server Programmer’s Guide for information.  
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.” |
## Error messages

<table>
<thead>
<tr>
<th>Message text (or text fragment)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_NOT_EXIST</td>
<td>Most commonly, this is because the server component cannot be instantiated or an instance is no longer available. Verify that:  &lt;ul&gt;&lt;li&gt;The specified component is installed in the specified EAServer Manager package.&lt;/li&gt;&lt;li&gt;The specified EAServer Manager package is installed in the server.&lt;/li&gt;&lt;li&gt;The Java class, Windows DLL, or UNIX shared library that implements the component is available.&lt;/li&gt;&lt;li&gt;If you are instantiating a Java component, the component’s skeleton class is available.&lt;/li&gt;&lt;/ul&gt;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. See “Mirror Cache tab component properties” in Chapter 25 of the EAServer Programmer’s Guide for more information. 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.</td>
</tr>
<tr>
<td>NullPointerException</td>
<td>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.</td>
</tr>
<tr>
<td>ObjectKey::init: &lt;servername&gt;.cycle create failed: No such file or directory</td>
<td>This error may be seen on server start-up. Causes include:  &lt;ul&gt;&lt;li&gt;Your installation does not have a repository, or&lt;/li&gt;&lt;li&gt;The repository is corrupt, or&lt;/li&gt;&lt;li&gt;Repository/Server directory is not writeable.&lt;/li&gt;&lt;/ul&gt;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.</td>
</tr>
<tr>
<td>SetDwObject (PBL_name, DataWindow_name) failed = -1</td>
<td>Verify the HTML DataWindow property in the DataWindow painter. Verify that the PBL (PowerBuilder library) or PBD (PowerBuilder dynamic library) is in the system PATH of the server machine.</td>
</tr>
<tr>
<td>srv_read_packet: Protocol error occurred: length in header (21536) more than packet size(512)</td>
<td>Verify that the listener port matches the client protocol.</td>
</tr>
<tr>
<td>Message text (or text fragment)</td>
<td>Explanation</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SRVLIB message: Net-Library routine net_listener (host:port) failed in srv_start_listeners</td>
<td>The listener is currently in use. Network listeners must be unique for the server machine. If the server runs as a Windows NT service, verify that you did not attempt to start another instance of the same server.</td>
</tr>
<tr>
<td>Network error: status = 23 - Net-Lib protocol driver call to register a listener failed</td>
<td>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.</td>
</tr>
</tbody>
</table>
| SystemException: CORBA::COMM_FAILURE | Possible reasons for this error include:  
• The server is down.  
• The server has run out of connections.  
• You did not specify the correct host name or listener address. |
| TRANSACTION_ROLLED_BACK | 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.  
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_ROLLED_BACK: Optimistic Concurrency Control” messages in the server log, try changing the persistence setting to Generated Class, then regenerate stubs and skeletons for the component. |

**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 EAServer Manager to verify the client’s certificate. If the client’s CA certificate is not in the client PKCS token, install it. |
| Trust verification failed. SSL protocol X.509 certificate chain is invalid. | Use EAServer Manager to verify that the client’s certificate chain is valid. |
| Trust verification failed. SSL protocol X.509 certificate has expired. | Use EAServer Manager 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. | Install the certificate of the CA that signed the client certificate in EAServer, using EAServer Manager, and mark it trusted. You must be the server administrator to do this. |
| Trust verification failed. SSL protocol CA certificate is untrusted. | Have the server administrator mark the client’s CA as trusted. |
System exceptions

System exception identifiers are predefined and listed in Table 5-2.

**Table 5-2: System exception identifiers**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jaguar/ClientException</td>
<td>An error occurred internally to the ActiveX proxy. For example, you</td>
</tr>
<tr>
<td></td>
<td>may have called a method that uses an unsupported parameter type.</td>
</tr>
<tr>
<td>CORBA/BAD_CONTEXT</td>
<td>Raised when a client invokes the operation but the passed context</td>
</tr>
<tr>
<td></td>
<td>does not contain the context values required by the operation.</td>
</tr>
<tr>
<td>CORBA/BAD_INV_ORDER</td>
<td>Indicates that the caller has invoked operations in the wrong order;</td>
</tr>
<tr>
<td></td>
<td>for example, you cannot make an ORB-related call without first</td>
</tr>
<tr>
<td></td>
<td>initializing the ORB.</td>
</tr>
<tr>
<td>CORBA/BAD_PARAM</td>
<td>A passed parameter is out of range or considered illegal, such as a</td>
</tr>
<tr>
<td></td>
<td>null values or null pointer.</td>
</tr>
<tr>
<td>CORBA/BAD_OPERATION</td>
<td>Indicates that an object reference denotes an existing object, but the</td>
</tr>
<tr>
<td></td>
<td>object does not support the operation that was invoked.</td>
</tr>
<tr>
<td>CORBA/BAD_TYPECODE</td>
<td>The datatype for a variable could not be determined or processed.</td>
</tr>
<tr>
<td>CORBA/COMM_FAILURE</td>
<td>A network error occurred. When creating a connection, this usually</td>
</tr>
<tr>
<td></td>
<td>indicates that the server is down or you have specified the wrong</td>
</tr>
<tr>
<td></td>
<td>listener address. When calling a method, the error may indicate a</td>
</tr>
<tr>
<td></td>
<td>transient network fault; you can retry the method.</td>
</tr>
<tr>
<td>Identifier</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BA/DATA_CONVERSION</td>
<td>A code set conversion problem occurred.</td>
</tr>
<tr>
<td>CORBA/FREE_MEM</td>
<td>JaguarORB failed in an attempt to free dynamic memory; for example, because of heap corruption or memory segments being locked.</td>
</tr>
<tr>
<td>CORBA/IMP_LIMIT</td>
<td>Indicates that an implementation limit has exceeded in the ORB runtime; for example, a parameter’s size may have exceeded the allowed maximum.</td>
</tr>
<tr>
<td>CORBA/INTERNAL</td>
<td>Indicates an internal failure in the JaguarORB, for example, if JaguarORB has detected corruption of its internal data structures.</td>
</tr>
<tr>
<td>CORBA/INTF_REPOS</td>
<td>Component information or properties could not be read from the repository.</td>
</tr>
<tr>
<td>CORBA/INV_FLAG</td>
<td>An invalid flag was passed to an operation.</td>
</tr>
<tr>
<td>CORBA/INV_IDENT</td>
<td>An RMI/IDL (remote method invocation/interface definition language) repository ID could not be mapped to a Java class.</td>
</tr>
<tr>
<td>CORBA/INV_OBJREF</td>
<td>An object reference is invalid.</td>
</tr>
<tr>
<td>CORBA/INVALID_TRANSACTION</td>
<td>A problem occurred while attempting to process a transaction.</td>
</tr>
<tr>
<td>CORBA/INITIALIZE</td>
<td>JaguarORB has encountered a failure during its initialization, such as failing to acquire networking resources or detecting a configuration error.</td>
</tr>
<tr>
<td>CORBA/MARSHAL</td>
<td>A request or reply from the network is structurally invalid. This error typically indicates an error in either the client-side or server-side runtime.</td>
</tr>
<tr>
<td>CORBA/NO_IMPLEMENT</td>
<td>The component does not implement the method that you called.</td>
</tr>
<tr>
<td>CORBA/NO_MEMORY</td>
<td>The ORB runtime has run out of memory.</td>
</tr>
<tr>
<td>CORBA/NO_RESOURCES</td>
<td>JaguarORB has encountered some general resource limitation; for example, the runtime may have reached the maximum permissible number of open connections.</td>
</tr>
<tr>
<td>CORBA/NO_RESPONSE</td>
<td>This exception is raised if a client attempts to retrieve the result of a deferred synchronous call, but the response for the request is not yet available.</td>
</tr>
<tr>
<td>CORBA/NO_PERMISSION</td>
<td>The user cannot access the server or a specified component.</td>
</tr>
<tr>
<td>CORBA/OBJ_ADAPTER</td>
<td>Indicates an administrative mismatch; for example, a server may have made an attempt to register itself using a name that is already in use, or is unknown to the repository.</td>
</tr>
</tbody>
</table>
## System exceptions

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORBA/OBJECT_NOT_EXIST</td>
<td>The object does not exist. This can happen if:</td>
</tr>
<tr>
<td></td>
<td>• The component installed incorrectly on the server. For example, the component class or skeleton class cannot be loaded.</td>
</tr>
<tr>
<td></td>
<td>• The object represents a stateful component and your reference to it has expired. Check the value of the component’s property, and, if</td>
</tr>
<tr>
<td></td>
<td>needed, code your client to create another instance in response to this error.</td>
</tr>
<tr>
<td>CORBA/PERSIST_STORE</td>
<td>A persistent storage failure; for example, failure to establish a database connection or corruption of a database.</td>
</tr>
<tr>
<td>CORBA/TRANSACTION_REQUIRED</td>
<td>The method you attempted to call must be called in the context of an open transaction.</td>
</tr>
<tr>
<td>CORBA/TRANSACTION_ROLLEDBACK</td>
<td>The method you called rolled back its transaction, or if you have started a client-managed transaction, the transaction timed out.</td>
</tr>
<tr>
<td>CORBA/TRANSIENT</td>
<td>An attempt to establish a connection fails, for example, because the server or the implementation repository is down.</td>
</tr>
<tr>
<td>CORBA/UNKNOWN</td>
<td>An operation implementation throws a non-CORBA exception or raises a user exception that does not appear in operation’s raises exception.</td>
</tr>
</tbody>
</table>

See the CORBA specification for general descriptions of CORBA system exceptions.

See the *EAServer Programmer's Guide* for examples of exception handling code.
CHAPTER 6

Advanced Topics

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational management tools</td>
<td>67</td>
</tr>
<tr>
<td>Debuggers</td>
<td>74</td>
</tr>
<tr>
<td>Stack traces, dump files, and core files</td>
<td>76</td>
</tr>
<tr>
<td>EAServer plug-in for JBuilder</td>
<td>78</td>
</tr>
<tr>
<td>Miscellaneous topics</td>
<td>81</td>
</tr>
</tbody>
</table>

**Operational management tools**

A number of tools and techniques are available 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 74.
- Other Windows tools are described under “Runtime monitoring tools” on page 69.

UNIX systems also provide a number of tools:

- See “UNIX debugging tools” on page 76.
- On HP-UX, numerous commands and tools are available including:
  - `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.
Operational management tools

- On Solaris, numerous commands are available, including:
  - `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:

```java
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 properties

Evaluating Windows memory


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

- How to monitor memory
- Interpreting output
- How to 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 the EAServer/PowerBuilder Memory Tuning and Troubleshooting at http://www.sybase.com/detail?1027319.

Runtime monitoring tools

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

**EAServer Manager**

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

**Debugging wizards**

These wizards step you through the configuration that enables tracing and debugging of components and servers. To run the wizard on a server, highlight the server icon, then choose File | Server Debug Settings Wizard. To run the wizard on a component, highlight the component icon, then choose File | Component Debug Settings Wizard.

**Performance tuning wizards**

These wizards guide you through the tuning of server, component, Web application, servlet/JSP or server settings that affect performance. To run them, highlight the server, component, Web application, or JSP icon, then choose File | Performance Tuning Wizard.

**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; you can 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:

- Chapter 12, “Using jagtool and jagant,” in the EAServer System Administration Guide
- Chapter 33, “Creating Service Components,” in the EAServer Programmer’s Guide

**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 jagsrv.exe >myfile.txt
```
Operational management tools

The following is sample output:

ListDLLs V2.23 - DLL lister for Win9x/NT
Copyright (C) 1997-2000 Mark Russinovich
http://www.sysinternals.com

jagsrv.exe pid: 330
Command line: jagsrv.exe Jaguar -c

Base Size Version Path
0x00400000 0x1f000 4.00.1381.0174 D:\Sybase\EAServer5\jagsrv.exe
0x77f60000 0x5e000 4.00.1381.0174 C:\WINNT\System32\ntdll.dll
0x10000000 0xae000 D:\Sybase\EAServer\lib\libjdispatch.dll
0x00230000 0x15000 D:\Sybase\EAServer\lib\libjaf.dll
0x00250000 0xc0000 D:\Sybase\EAServer\lib\libjintl.dll
0x77f00000 0x5e000 4.00.1381.0178 C:\WINNT\system32\KERNEL32.dll
0x77dc0000 0x3f000 4.00.1381.0203 C:\WINNT\system32\ADVAPI32.dll
0x77e70000 0x54000 4.00.1381.0133 C:\WINNT\system32\USER32.dll
0x77ed0000 0x100000 4.00.1381.0115 C:\WINNT\system32\GDI32.dll
0x78000000 0x43000 6.01.8293.0000 C:\WINNT\system32\MSVCRT.dll
0x00260000 0x59000 D:\Sybase\EAServer\lib\libjcomn.dll

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), use Process Explorer as follows:

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 javsrv.exe and jagsrvagent.exe processes to see which DLLs each 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
Profiling tools

Profiling tools allow you to identify and track performance issues.

For Java profiling, refer to the following documents:

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

- TechNote 1011550, Memory Management in 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 2000, 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.
SQL tracing

You can trace the SQL commands sent through connection caches. For more information, see “SQL tracing properties” in Chapter 4, “Database Access,” in the EAServer System Administration Guide.

Debuggers

Debuggers and crash handlers can:

• Generate and analyze stack dumps and core files (see “Stack traces, dump files, and core files” on page 76 for more information)
• 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 two Java debugging interfaces when you are running the debug version of the server. You can select the debugging interface on the Java Debug tab of the Server Properties dialog box:

• To use the JPDA interface, select Use JPDA, and specify the port.
• To use the sun.tools.debug interface, select Use Agent.

For details about Java debugging, see:

• Chapter 11, “Creating CORBA Java Components,” in the EAServer Programmer’s Guide, and
• Sun’s Java 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 that file is 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>
<thead>
<tr>
<th>Able to create dump file upon</th>
<th>AutoDump+</th>
<th>CDB</th>
<th>Dr. Watson</th>
<th>UserDump</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crash</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Exception</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Stack traces, dump files, and core files

<table>
<thead>
<tr>
<th>Able to create dump file upon</th>
<th>AutoDump+</th>
<th>CDB</th>
<th>Dr. Watson</th>
<th>UserDump</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hang</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Startup failure</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Normal run</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

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
These files are time-consuming to read and not always easy to understand. It is better to 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” 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 74)
- 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 read the dump file with various tools. For a comparison of the kinds of dump file analysis provided by the tools, see “Windows debugging tools” on page 74.

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 and 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 76.

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

**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 the Sybase white paper at http://www.sybase.com/detail?id=1028173.

❖ **Verifying your configuration**

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

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. Verify using 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 CLASSPATH.
   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 EAServer Manager or jagtool.
   2  Run the J2SDKEE Sun verifier on the generated EAR, JAR, or WAR file; for example, on Windows:
      ```
      verifier.bat myWAR.war
      ```

JBuilder JSPs that use connection caches to retrieve 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 the Sybase white paper at http://www.sybase.com/detail?id=1028828.

❖  Troubleshooting JSPs in JBuilder
   1  Verify your configuration—see “Verifying your configuration” on page 78.
   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:8080.
   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 Jaguarhttpservlet.log and Jaguar.log for error details.
   b Add an error page to the JSP, then rebuild and redeploy the project. For example:
      1 Add the following line to your JSP:
         <jsp:directive.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, you must rebuild the WAR file before redeploying it to EAServer; otherwise, the WAR file does not include your changes.
Miscellaneous topics

Additional tools and utilities

For more information about tuning and debugging tools and utilities, see the Tuning/Debug 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 316112 at http://support.microsoft.com/default.aspx?scid=kb;en-us;316112.

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
**Systems Management Web console**

To use the Systems Management Web console, you must enable JavaScript. If you use Netscape 7.0 with JavaScript disabled, you may not be able to navigate the Systems Management interface, and the wizards may not work correctly.

**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 *Jaguar.log*:

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

**Duplicate server names in EAServer Manager**

The server names listed under the *Servers* folder in EAServer Manager should be unique. If you see multiple servers with the same name, look in your $JAGUAR/Repository/Server directory. For each server that is defined, there is a `<servername>.props` file, and within each file, the name of the server is identified by the `com.sybase.jaguar.server.name` property. For example, in *Jaguar.props*, the following line defines the name of the Jaguar server:

```
com.sybase.jaguar.server.name=Jaguar
```

If more than one property file contains the same server name, you must edit the file, and change the value of `com.sybase.jaguar.server.name`. 
Index

A
accessibility features xii
ActiveX clients
  exception handling 49
  structure of an exception 50
  using an error handler 49
ActiveX components
  error handling 48
  logging messages from 4
ActiveX proxy error 64
Adaptive Server Anywhere
documentation ix
  installation location 37
adb, UNIX debugging tool 78
Admin mode
debugging 8
  server starts in 27
ADPlus, Windows debugging tool 75
advanced topics 67
AIX
  monitoring server operations 67
  signals 38
  stack traces, obtaining 9
Alert Management System 82
analyzing
  applications and environments 1
  information required 1
  stack dumps 68
Apache Log4j logging, integrating with 5
Apache TomCat servlet engine 25
application issues 28
  common problems 28
AutoDump+, Windows debugging tool 75
  avoiding memory leaks 32

B
BA/DATA_CONVERSION, system exception 65

BOOTCLASSPATH, environment variable 36

C
C++ components
  CORBA system exceptions 47
  error handling 47
  logging messages from 4
  memory corruption 30
  running externally 22, 30
  user-defined exceptions 48
C++ profiling 73
call statistics, logging 11
CDB, Windows debugging tool 75
certifications on the Web, Sybase xi
classes
  loading from unexpected locations 36
CLASSPATH environment variable 20, 36, 37, 64
ClassSearch utility 36
client applications
  performance issues 33
  client connections, limits 39
  Client Hotspot JVM, disabling JIT 13
  client log files for debugging 8
  client proxies 29
  cluster naming properties, logging 11
  CMP wrapper drivers
    connections, tracing 14
    stored procedures, tracing 14
    tracing 13
    transactions, tracing 13
code sets
  conversion error 65
  PowerBuilder clients and components 31
CodeXchange samples 81
  com.sybase.jaguar.CosNaming.debug, tracing property 11
  com.sybase.jaguar.instancepool.debug, tracing property 11
Index

com.sybase.jaguar.listener.http.connector_events, setting
for redirector 33
com.sybase.jaguar.security.logpeerIP tracing property 12
com.sybase.jaguar.server.dynamo.trace, tracing property 11
com.sybase.jaguar.server.flowcontrol.trace 11
com.sybase.jaguar.server.flowcontrol.trace, tracing property 11
com.sybase.jaguar.server.http.cache.debug, tracing property 11
com.sybase.jaguar.server.http.elffenable, tracing property 11
com.sybase.jaguar.server.http.elffitems, tracing property 11
com.sybase.jaguar.server.http.requestlogenable, tracing property 11
com.sybase.jaguar.server.iiop.log, tracing property 11
com.sybase.jaguar.server.iiop.log.ac, tracing property 11
com.sybase.jaguar.server.jvm.verbose property 21
com.sybase.jaguar.server.logspid, tracing property 10
com.sybase.jaguar.server.lwc.debug, tracing property 10
com.sybase.jaguar.server.traceattentions, MASP tracing 11
com.sybase.jaguar.server.tracenetdriver, Open Server tracing property 11
com.sybase.jaguar.server.tracenetrequests, Open Server tracing property 11
com.sybase.jaguar.server.tracetdsdata, MASP tracing property 11
com.sybase.jaguar.server.tracetdshdr, MASP tracing property 11
com.sybase.jaguar.threadmonitor.callstats, tracing property 10
com.sybase.jaguar.threadmonitor.trace, tracing property 10
common error messages 55
common problem areas 19
component skeletons, synchronizing 29
component.debug, debugging property 25
component.trace, tracing property 25
components
C++, running externally 30
invoking, problems 35
looping service 38
method not implemented, exception 65
shared 28
testing 25
tracing 12
user cannot access 65
concurrent client connection limits 39
configuration
issues 35
verifying 36
connecting
client, problems 35
connection caches
message service, using 82
testing 24
connections
failure, exception 66
network error creating 64
problems 28
references to 29
SSL, logging failed 12
tracing 13
console messages for debugging 8
container-managed persistence. See CMP context error 64
conventions x
CORBA system exceptions
C++ 47
Java 45
CORBA/BAD_CONTEXT, system exception 64
CORBA/BAD_INV_ORDER, system exception 64
CORBA/BAD_OPERATION, system exception 64
CORBA/BAD_PARAM, system exception 64
CORBA/BAD_TYPECODE, system exception 64
CORBA/COMM_FAILURE, system exception 64
CORBA/FREE_MEM, system exception 65
CORBA/IMP_LIMIT, system exception 65
CORBA/INITIALIZE, system exception 65
CORBA/INTERNAL, system exception 65
CORBA/INTF_REPOS, system exception 65
CORBA/INV_FLAG, system exception 65
CORBA/INV_IDENT, system exception 65
CORBA/INV_OBJREF, system exception 65
CORBA/INVALID_TRANSACTION, system exception 65
CORBA/MARSHAL, system exception 65
CORBA/NO_IMPLEMENT, system exception 65
CORBA/NO_MEMORY, system exception 65
84
### Index

<table>
<thead>
<tr>
<th>Issue</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORBA/NO_PERMISSION, system exception</td>
<td>65</td>
</tr>
<tr>
<td>CORBA/NO_RESOURCES, system exception</td>
<td>65</td>
</tr>
<tr>
<td>CORBA/NO_RESPONSE, system exception</td>
<td>65</td>
</tr>
<tr>
<td>CORBA/OBJ_ADAPTER, system exception</td>
<td>65</td>
</tr>
<tr>
<td>CORBA/OBJECT_NOT_EXIST, system exception</td>
<td>66</td>
</tr>
<tr>
<td>CORBA/PERSIST_STORE, system exception</td>
<td>66</td>
</tr>
<tr>
<td>CORBA/TRANSACTION_REQUIRED, system exception</td>
<td>66</td>
</tr>
<tr>
<td>CORBA/TRANSACTION_ROLLEDBACK, system exception</td>
<td>66</td>
</tr>
<tr>
<td>CORBA/TRANSIENT, system exception</td>
<td>66</td>
</tr>
<tr>
<td>CORBA/UNKNOWN, system exception</td>
<td>66</td>
</tr>
<tr>
<td>core dumps</td>
<td>8</td>
</tr>
<tr>
<td>core files</td>
<td>76, 78</td>
</tr>
<tr>
<td>could not start thread, error message</td>
<td>58</td>
</tr>
<tr>
<td>CPU</td>
<td></td>
</tr>
<tr>
<td>processes consuming</td>
<td>68</td>
</tr>
<tr>
<td>sizing</td>
<td>38</td>
</tr>
<tr>
<td>usage, determining</td>
<td>68</td>
</tr>
<tr>
<td>crash address, finding</td>
<td>22</td>
</tr>
<tr>
<td>crashing, servers</td>
<td>20</td>
</tr>
<tr>
<td>Ctrl+Break, to obtain stack trace</td>
<td>9</td>
</tr>
<tr>
<td>ADPlus</td>
<td>75</td>
</tr>
<tr>
<td>AutoDump+</td>
<td>75</td>
</tr>
<tr>
<td>CDB</td>
<td>75</td>
</tr>
<tr>
<td>dbx</td>
<td>76</td>
</tr>
<tr>
<td>Dr. Watson</td>
<td>75</td>
</tr>
<tr>
<td>gdb</td>
<td>76</td>
</tr>
<tr>
<td>jdb</td>
<td>74</td>
</tr>
<tr>
<td>kd</td>
<td>75</td>
</tr>
<tr>
<td>sdb</td>
<td>78</td>
</tr>
<tr>
<td>UserDump</td>
<td>75</td>
</tr>
<tr>
<td>WinDbg</td>
<td>75</td>
</tr>
<tr>
<td>deploying EAR, JAR, or WAR files</td>
<td>29</td>
</tr>
<tr>
<td>disappearing servers, investigating</td>
<td>26</td>
</tr>
<tr>
<td>disconnecting client proxies</td>
<td>29</td>
</tr>
<tr>
<td>DLL monitoring</td>
<td>71</td>
</tr>
<tr>
<td>documentation, where to find</td>
<td>vii</td>
</tr>
<tr>
<td>Dr. Watson, Windows debugging tool</td>
<td>75</td>
</tr>
<tr>
<td>drivers using the DataSource interface</td>
<td>81</td>
</tr>
<tr>
<td>dump files</td>
<td>76</td>
</tr>
<tr>
<td>Windows</td>
<td>77</td>
</tr>
<tr>
<td>duplicate server names</td>
<td>82</td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>EAR files, deploying</td>
<td>29</td>
</tr>
<tr>
<td>EAServer</td>
<td></td>
</tr>
<tr>
<td>attaching a debugger to</td>
<td>76</td>
</tr>
<tr>
<td>cannot start</td>
<td>37</td>
</tr>
<tr>
<td>connection problems</td>
<td>28</td>
</tr>
<tr>
<td>documentation</td>
<td>vii</td>
</tr>
<tr>
<td>log file</td>
<td>4</td>
</tr>
<tr>
<td>monitoring APIs</td>
<td>70</td>
</tr>
<tr>
<td>plug-in for JBuilder</td>
<td>78</td>
</tr>
<tr>
<td>service, running as a</td>
<td>37</td>
</tr>
<tr>
<td>starting, trouble</td>
<td>35</td>
</tr>
<tr>
<td>traces</td>
<td>10</td>
</tr>
<tr>
<td>EAServer Manager</td>
<td></td>
</tr>
<tr>
<td>client host and thread, monitoring</td>
<td>70</td>
</tr>
<tr>
<td>HTTP and IIOP network monitoring</td>
<td>70</td>
</tr>
<tr>
<td>HTTP monitoring</td>
<td>17</td>
</tr>
<tr>
<td>IIOP monitoring</td>
<td>17</td>
</tr>
<tr>
<td>tracing thread usage</td>
<td>70</td>
</tr>
<tr>
<td>EAServer plug-in for JBuilder X</td>
<td></td>
</tr>
<tr>
<td>debugging</td>
<td>78</td>
</tr>
<tr>
<td>deploying externally</td>
<td>79</td>
</tr>
</tbody>
</table>

### Troubleshooting Guide

- CORBA/NO_PERMISSION, system exception 65
- CORBA/NO_RESOURCES, system exception 65
- CORBA/NO_RESPONSE, system exception 65
- CORBA/OBJ_ADAPTER, system exception 65
- CORBA/OBJECT_NOT_EXIST, system exception 66
- CORBA/PERSIST_STORE, system exception 66
- CORBA/TRANSACTION_REQUIRED, system exception 66
- CORBA/TRANSACTION_ROLLEDBACK, system exception 66
- CORBA/TRANSIENT, system exception 66
- CORBA/UNKNOWN, system exception 66
- core dumps 8
- core files 76, 78
- could not start thread, error message 58
- CPU processes consuming 68
- sizing 38
- usage, determining 68
- crash address, finding 22
- crashing, servers 20
- Ctrl+Break, to obtain stack trace 9

D

- data sources, ODBC 37
- DataSource interface, drivers using 81
- DBMS error, PowerBuilder 59
- dbparam, transaction property 36
- dbx, UNIX debugging tool 76
- debuggers 74
- debugging
  - attaching a debugger to EAServer 76
  - EAServer plug-in for JBuilder X 78
  - Java 74
  - JSPs 52
  - UNIX 76
  - Windows 74
- debugging aids
  - client log files 8
  - server console messages 8
  - wizards 70
- debugging tools
  - adb 78

E

- EAR files, deploying 29
- EAServer
  - attaching a debugger to 76
  - cannot start 37
  - connection problems 28
  - documentation vii
  - log file 4
  - monitoring APIs 70
  - plug-in for JBuilder 78
  - service, running as a 37
  - starting, trouble 35
  - traces 10
- EAServer Manager
  - client host and thread, monitoring 70
  - HTTP and IIOP network monitoring 70
  - HTTP monitoring 17
  - IIOP monitoring 17
  - tracing thread usage 70
- EAServer plug-in for JBuilder X
  - debugging 78
  - deploying externally 79
Index

generating verbose logging 79
reviewing deployment output 79
verifying your configuration 78
EBFs and software maintenance xi
ECHO property 21
EJBs using lightweight container 10

environment variables
BOOTCLASSPATH 36
CLASSPATH 20, 36, 37, 64
JAGUAR 37
JAGUAR_CLIENT_ROOT 64
JAGUAR_RANDOMSEED 20
LD_LIBRARY_PATH 37
PATH 20, 37, 64
PBOnFatalError 54
PBRollbackOnRTErr 30
required for debugging 2
SOLANY 37
THREADS_FLAG 37
WSPLUGIN_CONFIG_FILE 34

error handling
ActiveX components 48
C++ components 47
Java components 44
lack of 29
PowerBuilder 54

error messages
common 55
could not start thread 58
message text and explanations 56
source indicators 55
system exception identifiers 64

error pages
adding for Web applications 51
for JavaServer Pages 51
JSP sample 52
server-level 50
using JSPs for 52
for Web applications 51

evaluating Windows memory 69
exception handling 43
ActiveX clients 49
Java clients 44
exceptions
PowerBuilder, unhandled 54

F
file descriptors 39
listing 68
flow control, logging 11

G
gdb. UNIX debugging tool 76
getserverinfo, securetool server status monitoring command 71
getservicestate, securetool service component monitoring command 71
Getting Started CD x

H
handling exceptions 43
hanging, servers 20, 24
HP-UX commands
top 67
tusc 67
vmstat 67

HTTP
and IIOP network monitoring using EAServer Manager 70
error log 7
error logs 2
network tracing tools 17
request log 7
request logging, properties to enable 24
requests, logging 11
servlet log 7

I
IIOP
and HTTP network monitoring using EAServer Manager 70
log file 7
logging and slow performance 7
network tracing tools 17
IIS redirector plug-in 33
imagecfg, Windows utility 39

86

EAServer
Index

improving performance 41
information required for Technical Support 1
installation issues 19
installation log file, generating 8
Internet Explorer
  security patch, problems 81
invalid object reference 65
isql tool 24

J
JagRepair server, using 21
jagsleepl command 28
jagsrv process 26
jagtool monitoring commands 71
JAGUAR environment variable 37
Jaguar/ClientException, system exception 64
Jaguar::PerfMonitor API 71
Jaguar::StatProvider API 71
Jaguar_boot.log file 8
JAGUAR_CLIENT_ROOT environment variable 64
JAGUAR_RANDOMSEED environment variable 20
JaguarORB
  error freeing memory 65
  initialization failure 65
  internal error 65
  resource limitation exception 65
JAR files, deploying 29
Java
  class loading, tracing 13
  debugging tools 74
  JVM properties, logging 13
  predebugging, in-server 13
  profiling 73
  static variables 29
  VM options property, setting 13
Java clients
  CORBA system exceptions 45
  exception handling 44
  user-defined exceptions 46
Java components
  error handling 44
  logging messages from 4
Java Logging system APIs 6
Java Transaction API, tracing 14
JavaScript and the Web console 82
JavaServer Pages
  See also JSP
  error pages for 51
  uncaught exceptions 52
  using as an error page 52
JBuilder X
  EAServer plug-in for 78
  JSPs, debugging 79
jdbc, Java debugging tool 74
JDK 1.4 Java Logging package, integrating with 5
JIT, disabling 13
JPDA 74
JSP
  See also JavaServer Pages
  error page, sample 80
  in JBuilder X 79
  saving source code 52
JVM options property, setting 13
JVM properties, logging 13

K
kd, Windows debugging tool 75
keyboard navigation xii

L
LD_LIBRARY_PATH environment variable 37
libjcc.dll file problems 36
libraries, determining which are loaded 68
lightweight container, logging EJB calls that use 10
ListDLLs, freeware for Windows 71
listeners
  tracing 11
  verify port availability 36
loading classes 36
loads, peak 38
log files
  EAServer 4
  HTTP errors 7
Index

HTTP requests 7
HTTP servlet 7
IIOP 7
installation 8
Log Profile 5
log profiles, managing 6
Log4j logging APIs 6
logging 4
See also tracing
APIs 6
call statistics 11
flow control 11
HTTP requests 11
IIOP 11
integrating with other systems 5
JVM properties 13
named instance pools 11
peak number of threads 10
PowerDynamo plug-in calls 11
server and cluster naming properties 11
server messages 4
synchronization 11
table statistics for CMP 14
thread identifiers 10
looping service components 38

M
managing log profiles 6
MASP
tracing 11
using to check TDS listener 24
maximum threads property 58
memory
avoiding leaks 32
corruption in C++ components 30
drain, ResultSets 28
lack of 27
management tools 68
PBVM, tuning for 30
usage sample 68
usage, determining 68
memory leaks, diagnosing
OptimizeIt 27
Performance Monitor 27

Message Bridge for Java documentation ix
message service
connection cache 82
tracing 12
Methods As Stored Procedures. See MASP
miscellaneous topics 81
monitoring
client host and thread information 70
DLLs 71
techniques 1
monitoring tools 67
EAServer monitoring APIs 70
Java profiling 73
ListDLLs 71
Process Explorer 72
ps, UNIX command 73
Purify 73
PurifyPlus 73
multiprocessor issues 38

N
named instance pools, logging 11
netstat, network tracing tool 17
network driver requests, tracing 11
network tracing tools
HTTP 17
IIOP 17
TCP/IP 17
TDS 18
networks, invalid replies from 65
non-threadsafe sleep calls 28

O
object reference invalid 65
object reference, invalid 65
objects
do not exist, exception 66
unsupported operation exception 64
obtaining stack traces 9
ODBC data sources 37
Open Server tracing 11
operating system
patches 38
signal issues 8
operational management tools 67
operations, invoking in the wrong order 64
OptimizeIt, tool for diagnosing memory leaks 27
ORB runtime
implementation limit error 65
out of memory error 65
-ORBCodeSet property 31
overview
exception handling 43
monitoring 1

P
parameters
illegal 64
out of range 64
patches, operating system 38
PATH environment variable 20, 37, 64
PBOntalError variable 30, 54
PBRollbackOnRTErrovariable 30
PBVM
memory tuning 30
and PowerBuilder versions 36
peak loads 38
performance
client application issues 33
IIOP logging and server 7
issues 41
tuning 41
Performance Monitor, using to diagnose memory leaks 27
performance tuning wizards 70
persistent storage failure 66
pfies, UNIX command 68
ping, network tracing tool 17
pldd, UNIX command 68
pmap, UNIX command 20, 68
pmon, Windows command 73
pooled components 29
PowerBuilder
code sets 31
components, logging messages from 4
DBMS error 59
error handling 54
PBOntalError variable 30
PBRollbackOnRTErrovariable 30
and PBVM versions 36
unhandled exceptions 54
Web DataWindows 30
PowerDynamo calls, logging 11
Process Explorer, monitoring tool 72
processes
finding active 73
tools used to monitor 73
proexp.exe, Process Explorer program file 72
production environment, verify settings 36
profiling
C++ 73
Java 73
tools 73
protocol IIOPS not supported, warning 64
ps, UNIX command 73
PsList, Windows freeware 73
pstack, UNIX command 68
psstat, Windows command 73
Purity, profiling tool for UNIX 73
PurityPlus, profiling tool for Windows 73

R
redirecting server output 26
redirector plug-in issues 33
redirector.cfg configuration file 34
repository
reading from, error 65
resource allocation problems associated with service 37
resources
performance tuning 41
ResultSets
memory drain 28
runtime monitoring
processes 73
tools 69
Index

S
samples
   exceptions, trapping in C++ 47
   JSP error page 52, 80
   memory usage and JVM properties 68
sdb, UNIX debugging tool 78
server console messages for debugging 8
server failures 20
   cluster members 20
   crash address, finding 22
   disappears 26
   intermittently 23
   on start-up 20
   reproducible 22
server output, redirecting 26
server performance and IIOP logging 7
server-level
   error pages 50
   tracing 10
servers
   Admin mode, starting in 27
   cannot start 37
   crashing 20
   diminishing performance 27
   duplicate names 82
   hanging 24
   naming properties, logging 11
   problems connecting to 28
   processes, records of 8
   running out of memory 27
   user cannot access 65
service, running EAServer as 37
servlets, tracing 12
session cookies, problems using IE 81
shared components 28
signal issues, operating system 8
sleep calls, non-threadsafe 28
sockets, listing 68
Solaris timers 38
SQL, tracing commands 74
SQLANY environment variable 37
SSL
   CORBA::NO_PERMISSION exception 57
   logging failed connections 12
   Service Provider error 64
   X.509 certificate chain error 63
stack dumps, analyzing 68
stack traces 76
   obtaining 9
   using 8
stateful components
   client proxy not disconnected 29
static page caching, tracing 11
static variables in Java 29
Synchronization -verbose, tracing command 11
synchronization, logging 11
synchronizing component skeletons 29
synchronous call, unavailable response exception 65
system exceptions 64
system-level issues 38
Systems management Web console 82

T
table statistics for CMP, logging 14
Tabular Data Stream. See TDS 18
TCP/IP, network tracing tools 17
TDImon, network tracing tool 17
TDS
   checking listener ports 24
   header contents, tracing 11
   network tracing tools 18
   packet contents, tracing 11
   protocol attentions, tracing 11
Technical Library CD x
Technical Support, information required for 1
testing
   components 25
   connection caches 24
threads
   could not start, error message 58
   locking 38
   logging identifiers 10
   maximum 58
   user-spawned 28
THREADS_FLAG environment variable 37
TomCat servlet engine 25
tools
   memory management 68
   monitoring 67
   Optimizel 27
Performance Monitor 27
profiling 73
runtime monitoring 69
top
HP-UX command 67
UNIX command 68
traceroute, network tracing tool 17
tracing
See also logging
CMP wrapper drivers 13
components 12
components and server 70
connections 13
connections for CMP wrapper drivers 14
Java class loading 13
Java in-server predebugging 13
Java Transaction API 14
listeners 11
MASP 11
message service 12
network driver requests 11
server-level 10
servlets 12
SQL commands 74
static page caching 11
stored procedures for CMP wrapper drivers 14
TDS header contents 11
TDS packet contents 11
TDS protocol attentions 11
thread usage with EAServer Manager 70
transactions 13
transactions for CMP wrapper drivers 14
transport control layer 11
Web applications 12
Web server redirector plug-ins 14
transactions
dbparam property 36
required but missing, exception 66
rolled back or timed out, exception 66
system exception 65
tracing 13
transport control layer, tracing 11
truss, UNIX command 68
tuning
application performance 41
components, servers, and Web applications 70
tuning and debugging, CodeXchange samples 81
tusc, HP-UX command 67
typographical conventions x
U
ulimit, command 39
unable to initialize the VM, error message 64
unhandled PowerBuilder exceptions 54
UNIX
core files 78
debugging tools 76
file descriptors 39
unknown datatype exception 64
user-defined exceptions
C++ 48
Java 46
UserDump, Windows debugging tool 75
user-spawned threads 28
using error pages 50
V
variables
datatype, unknown 64
holding references 29
network error 64
verifying configuration 36
virtual bytes, Windows limit 39
virtual machine, unable to initialize, error message 64
vmstat
HP-UX command 67
UNIX command 68
W
WAR files, deploying 29
warnings
failed to initialize SSL Service Provider 64
protocol IIOPS not supported 64
Web applications
adding an error page 51
deploying, problems 29
Index

error pages for 51
session cookies, problems using IE 81
testing 25
tracing 12
Web console, enabling JavaScript for 82
Web DataWindows, PowerBuilder 30
Web server redirector plug-ins
   com.sybase.jaguar.listener.http.connector_events
      property 33
   issues 33
   Microsoft IIS 33
   tracing 14
WinDbg, Windows debugging tool 75
Windows
   debugging tools 74
dump files 77
   memory, evaluating 69
   virtual bytes 39
wizards
   debugging 70
   performance tuning 70
WSPLUGIN_CONFIG_FILE environment variable 34

X
xterm utility 37