Feature Guide

EAServer
Version 5.2
DOCUMENT ID: DC38033-01-0520-01

LAST REVISED: January 2005

Copyright © 1997-2005 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.


Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

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

About This Book ............................................................................................................................ v

CHAPTER 1 Introducing EAServer ................................................................. 1
  Overview ................................................................. 2
  EAServer execution engine ......................................... 3
  Component support .................................................... 4
  J2EE platform support .............................................. 5
  Network protocol support ......................................... 7
  Dynamic HTML support ........................................... 9
    PowerDynamo Web site conversion utility ............... 10
    Web server redirector plug-in ............................... 10
  Administration and development tools .................. 10
  Client-session and component-lifecycle management .. 14
  Naming services ......................................................... 16
  Connection caching ..................................................... 18
  Transaction management .......................................... 18
  Localization ............................................................... 18
  Result-set support ...................................................... 20
  Support for asynchronous messaging .................... 21
  Asynchronous processing support ......................... 22
  Systems management support ................................. 22
  Support for legacy Open Server applications ........... 23
  IPV6 support .............................................................. 23

CHAPTER 2 Developing an EAServer Application ........................................... 25
  Introductory concepts ............................................... 25
  Developing an EAServer application ....................... 27
  EAServer runtime environment .............................. 28

CHAPTER 3 EAServer Components ................................................................. 31
  Overview ................................................................. 31
  Enterprise JavaBeans components ......................... 35
    EJB component types ............................................. 36
Contents

EJB transaction attribute values ................................................. 38
EJB container services .............................................................. 38
Java components ........................................................................... 39
CORBA-C++ components .............................................................. 40
PowerBuilder components ............................................................. 40
ActiveX components ...................................................................... 41
C components ................................................................................ 41

CHAPTER 4  Web Applications ................................................................. 43
What is a Web application? ............................................................ 43
Contents of a Web application ....................................................... 44
Servlet files .............................................................................. 44
JSP files and tag libraries ........................................................ 44
Static files ................................................................................ 45
Java classes ............................................................................ 45
Deployment descriptor ............................................................ 46

CHAPTER 5  Using PowerDynamo with EAServer ................................. 47
EAServer and PowerDynamo architecture ..................................... 48
Calling EAServer components from PowerDynamo scripts .... 49
Setting up PowerDynamo as a client ............................................. 51

CHAPTER 6  Using Message Bridge for Java with EAServer ................ 53
Message Bridge overview .......................................................... 53
Using Message Bridge ................................................................ 54
Message Bridge and EAServer architecture ................................. 54
For more information ............................................................... 56

Index .................................................................................................. 57
About This Book

Subject
This book describes EAServer, which is an integrated set of application servers that you use to deploy Web applications that support high-volume traffic, dynamic content, and intensive online transaction processing. The EAServer product set consists of PowerDynamo™ and Adaptive Server® Anywhere.

Note Products described in this manual may not be available on some UNIX platforms. See the Release Bulletin and the Installation Guide for a list of products included in your EAServer edition.

Audience
This book is written for new users of EAServer.

How to use this book
The following chapters are included in this book:

- Chapter 1, “Introducing EAServer,” includes a description of EAServer features and use.
- Chapter 2, “Developing an EAServer Application,” explains some of the basic concepts and terminology associated with developing component-based EAServer applications.
- Chapter 3, “EAServer Components,” discusses developing EJB, Java, C/C++, and ActiveX components that run on EAServer.
- Chapter 4, “Web Applications,” describes the components of Web applications.
- Chapter 6, “Using Message Bridge for Java with EAServer,” describes how Message Bridge is used to build applications that make use of structured messages, such as XML documents or messages exchanged between enterprise systems or business partners through New Era of Networks adapters, using EAServer components.
Related documents

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

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

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

The EAServer Feature Guide (this book) 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 Security Manager plug-in for Sybase Central

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

Implement secure HTTP and IIOP client applications

Deploy client applications that connect through Internet proxies and firewalls

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


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

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


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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>commands and methods</strong></td>
</tr>
<tr>
<td>When used in descriptive text, this font indicates keywords such as:</td>
</tr>
<tr>
<td>• Command names used in descriptive text</td>
</tr>
<tr>
<td>• C++ and Java method or class names used in descriptive text</td>
</tr>
<tr>
<td>• Java package names used in descriptive text</td>
</tr>
<tr>
<td>• Property names in the raw format, as when using jagtool to configure applications rather than EAServer Manager</td>
</tr>
<tr>
<td><strong>variable, package, or component</strong></td>
</tr>
<tr>
<td>Italic font indicates:</td>
</tr>
<tr>
<td>• Program variables, such as <em>myCounter</em></td>
</tr>
<tr>
<td>• Parts of input text that must be substituted, for example:</td>
</tr>
<tr>
<td><em>Server.log</em></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>
<tr>
<td>**File</td>
</tr>
<tr>
<td>Menu names and menu items are displayed in plain text. The vertical bar shows you how to navigate menu selections. For example, File</td>
</tr>
<tr>
<td><strong>package 1</strong></td>
</tr>
<tr>
<td>Monospace font indicates:</td>
</tr>
<tr>
<td>• Information that you enter in EAServer Manager, a command line, or as program text</td>
</tr>
<tr>
<td>• Example program fragments</td>
</tr>
<tr>
<td>• Example output fragments</td>
</tr>
</tbody>
</table>

### Other sources of information

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

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

- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format. Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.
Refer to the SyBooks Installation Guide on the Getting Started CD, or the README.txt file on the SyBooks CD for instructions on installing and starting SyBooks.

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

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

Sybase certifications on the Web

Finding the latest information on product certifications
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.
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. If prompted, enter your MySybase user name and password.
3. Select a product.
4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

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

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

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

EAServer Manager supports working without a mouse. For more information, see “Keyboard navigation” in Chapter 2, “Sybase Central Overview,” in the EAServer System Administration Guide.

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

1 Start Eclipse
2 Select Help | Help Contents
3 Enter Accessibility in the Search dialog box
4 Select Accessible user interfaces or Accessibility features for Eclipse

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

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

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.
CHAPTER 1

Introducing EAServer

This chapter describes the EAServer features.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>2</td>
</tr>
<tr>
<td>EAServer execution engine</td>
<td>3</td>
</tr>
<tr>
<td>Component support</td>
<td>4</td>
</tr>
<tr>
<td>J2EE platform support</td>
<td>5</td>
</tr>
<tr>
<td>Network protocol support</td>
<td>7</td>
</tr>
<tr>
<td>Dynamic HTML support</td>
<td>9</td>
</tr>
<tr>
<td>Administration and development tools</td>
<td>10</td>
</tr>
<tr>
<td>Client-session and component-lifecycle management</td>
<td>14</td>
</tr>
<tr>
<td>Naming services</td>
<td>16</td>
</tr>
<tr>
<td>Connection caching</td>
<td>18</td>
</tr>
<tr>
<td>Transaction management</td>
<td>18</td>
</tr>
<tr>
<td>Localization</td>
<td>18</td>
</tr>
<tr>
<td>Result-set support</td>
<td>20</td>
</tr>
<tr>
<td>Support for asynchronous messaging</td>
<td>21</td>
</tr>
<tr>
<td>Asynchronous processing support</td>
<td>22</td>
</tr>
<tr>
<td>Systems management support</td>
<td>22</td>
</tr>
<tr>
<td>Support for legacy Open Server applications</td>
<td>23</td>
</tr>
<tr>
<td>IPV6 support</td>
<td>23</td>
</tr>
</tbody>
</table>

For a description of the features that are new in this version, see What’s New in EAServer.
Overview

EAServer is an application server, which includes an integrated set of development tools that you use to deploy Web applications that support high-volume traffic, dynamic content, and intensive online transaction processing. The EAServer product set includes PowerDynamo and Adaptive Server Anywhere.

EAServer provides a framework for deploying the middle-tier logic of distributed component-based applications.

EAServer simplifies the creation and administration of Internet applications that service thousands of clients simultaneously. EAServer components execute on the middle-tier between end-user client applications and remote databases. EAServer provides efficient management of client sessions, security, threads, third-tier database connections, and transaction flow, without requiring specialized knowledge on the part of the component developer.

EAServer’s scalability and platform independence allow you to develop your application on inexpensive single-processor machines, then deploy the application on an enterprise-grade multiprocessor server.

EAServer provides the following features:

- A scalable, multithreaded, platform-independent execution engine
- Dispatch and stub/proxy support for all major component models, including JavaBeans, PowerBuilder®, Java, ActiveX, and C/C++
- Dynamic HTML support using Java servlet, JavaServer Pages, and PowerDynamo Web sites
- Java 2 Enterprise Edition (J2EE) platform support
- Graphical administration with Sybase Central, including component interface browsing, declarative role-based security, password, and required SSL session characteristics, server and user certificate management, IDL module support, transaction monitoring, and runtime monitoring
- Tight integration with the PowerBuilder development environment
- Transparent client-session and component-lifecycle management
- Connection caching to allow reuse of remote database connections
- Industry-standard naming services to resolve components using logical names rather than server addresses
CHAPTER 1  Introducing EAServer

• Transaction management to simplify the design and implementation of an application’s transactions
• Transparent thread-safety features to simplify use of shared data and resources
• Result-set support to enable efficient retrieval of tabular data in client applications
• Declarative, role-based security to restrict client connections and the components that can be invoked by a specific client session
• Identity-based security to restrict intercomponent calls
• Asynchronous messaging support
• Asynchronous processing support
• Web server redirector plug-in forwards client requests directly to a Web server
• EAServer provides a plug-in for Borland JBuilder version 10. The EAServer plug-in conforms to Section 508 guidelines, and is internationalized and localizable.

The following sections explain these features in detail.

EAServer execution engine

The EAServer runtime engine provides a scalable and platform-independent environment for the execution of component based applications. EAServer is scalable because it is multithreaded and multiprocessor safe.

The EAServer execution environment is the same across all platforms, with the exception of ActiveX component support. ActiveX requires platform support and is currently available only on Windows.

The EAServer runtime engine provides the following services:

• Network listeners for the connections on which clients send remote component invocations. EAServer core network server technology is based on Sybase Open Client/Server™ technology. “Network protocol support” on page 7 details the supported application protocols.
• An execution environment for middle-tier components. See “Server-side component support” on page 31.
Component support

- A built-in HTTP server. You can use EAServer HTTP support to deploy your application’s Java applets and HTML pages.
- Hosting of PowerDynamo Web sites so that you can access those sites from a browser.
- Ability to run with different Java virtual machines.
- A JagRepair server that you can use to repair your application server if you cannot start it up because of an incorrect configuration setting.
- Declarative security. You can use roles to authenticate and authorize users. See the EAServer Security Administration and Programming Guide.
- Connection caching. You can define caches of connections for interacting with remote databases from EAServer components. See “Connection caching” on page 18.
- Open Server™ event handler support. See “Support for legacy Open Server applications” on page 23.

In addition to these built-in services, you can install service components that run in the background and provide customized services to clients or other components. See Chapter 33, “Creating Service Components,” in the EAServer Programmer’s Guide.

Component support

Components are reusable modules of code that combine related tasks (methods) into a well-defined interface. EAServer components are installed on an EAServer application server and contain the methods that execute business logic and access data sources. You or your administrator install the component’s executable code on the server. Components can be distributed throughout a network, including the Internet or an intranet, on different servers. Installed components can be used by any number of independent applications.

Since EAServer components reside on the server, components do not contain methods to display graphics or user interfaces—that is, EAServer components are inherently nonvisual.
User-interface developers or other component developers can browse a component’s interface in EAServer Manager; in their code, they use a client stub or proxy to invoke the component’s methods. The stub or proxy acts as a local surrogate for the remote component, providing the same method signatures as the component and hiding the details of remote server communication.

The EAServer server-side component support and client-side stub or proxy support are independent. Any EAServer client can execute any type of component. A component of any model can execute components of another model using intercomponent calls without the use of additional gateway software. Additionally, since EAServer uses standard CORBA IIOP as its core network protocol, you can use CORBA client runtimes from other vendors to invoke components installed on EAServer.

All clients and components share a common interface repository. Component interfaces are stored in standard CORBA interface definition language (IDL). Component developers can define, edit, and browse interfaces in EAServer Manager, which allows you to edit interfaces graphically or as raw IDL. You can also define interfaces by importing compiled Java classes, standard-format EJB-JAR files, or ActiveX type libraries.

For more information about the component models that EAServer supports, see Chapter 3, “EAServer Components.”

J2EE platform support

EAServer implements the Java 2 Enterprise Edition (J2EE) 1.3 specification, with support for EJB 2.0 components, J2EE applications, J2EE Web applications, object caching, the JavaMail electronic mail API, the connector architecture, Java API for XML Parsing, and the Java Authentication and Authorization Services.

“Component support” on page 4 describes EAServer’s supported component models, including EJB.

A Web application is a unit of deployment for interrelated Web content, JavaServer Pages (JSPs), and Java servlets. The Web application contains static files, servlet and JSP implementation classes, and a deployment descriptor that describes how the files, servlets, and JSPs are configured on the host server. See Chapter 21, “Creating Web Applications,” in the EAServer Programmer’s Guide.
**J2EE platform support**

J2EE applications allow you to group related EJB 2.0 components and Web applications into a single entity. In this way, you can deploy related business logic components, Java servlets, JavaServer Pages, and Web pages as a single unit between servers. Using the J2EE Application Client model, you can create clients that call the components and Web pages in the application. For more information, see these chapters in the *EAServer Programmer’s Guide*:

- Chapter 3, “Managing Applications and Packages in EAServer Manager”
- Chapter 10, “Creating Application Clients”

EAServer supports EJB 2.0 container-managed persistence (CMP) for EJB entity bean components, as well as supporting an automatic persistence model for components of other types. EAServer also supports entity instance and finder-query caching, which can improve performance by minimizing the number of database select queries required to execute business logic. For information on entity components, automatic persistence, and object/query caching, see Chapter 27, “Creating Entity Components,” in the *EAServer Programmer’s Guide*.

The JavaMail API provides a standard Java interface to the most widely-used Internet mail protocols. See Chapter 35, “Creating JavaMail,” in the *EAServer Programmer’s Guide* for more information.

The J2EE connector architecture enables you to write portable Java applications that can access multiple transactional enterprise information systems. A connector is a specialized connection factory that provides connections for EJBs, Java servlets, JSPs, and CORBA-Java components. For more information, see Chapter 4, “Database Access,” in the *EAServer System Administration Guide*.

EAServer can host Web applications in popular Web servers such as Apache, iPlanet, and Netscape. For more information, see the *EAServer Installation Guide* for your platform.

EAServer implements J2EE version 1.3 security requirements including Java and C++ ORB support and CORBA Secure Interoperable version 2 protocol (CSIv2). For more information, see the *EAServer Security Administration and Programming Guide*.

EAServer includes support for the Java API for XML Parsing 1.1. You can configure the parser and transformer implementations for servers, components, Web applications, and application clients.
Java Authentication and Authorization Services (JAAS) provide a framework and standard programming interface for authenticating users and assigning privileges. JAAS is based on the Pluggable Authentication Module standard, which extends the access-control architecture of the Java 2 platform to support user-based authentication and authorization. For more information, see the EAServer Security Administration and Programming Guide.

Network protocol support

EAServer supports the following protocols:

- **Internet Inter-ORB Protocol (IIOP)**  
  IIOP is the standard protocol for communication between CORBA ORBs over TCP/IP networks. All EAServer client models except MASP use IIOP or IIOP tunnelled inside of SSL (referred to as IIOPS). IIOP connections can also be tunnelled inside of HTTP to allow connections through firewalls that do not allow passage of IIOP traffic, as discussed in “HTTP tunnelling support” on page 8.

- **Sybase Tabular Data Stream™ (TDS)**  
  TDS is a proprietary protocol used in two-tier database applications that connect to Sybase database servers or gateways. Two types of clients connect to EAServer using TDS:
  - **MASP**  
    MASP and TDS allow you to incorporate EAServer components into applications that were developed with traditional two-tier development tools.
  - **Legacy Open Server clients**  
    If you have converted an Open Server application to run in EAServer, legacy clients for the application connect to EAServer using TDS.

  To separate MASP and Open Server requests, EAServer requires different listener ports for each type of client. To support MASP clients, your server must have at least one TDS listener installed. You must define an Open Server listener to support legacy Open Server clients.

- **Hypertext Transfer Protocol (HTTP)**  
  HTTP is used by Web browsers for file downloads and uploads. EAServer provides HTTP support to allow you to deploy HTML pages and Java applets on EAServer itself.

- **Secure Sockets Layer (SSL)**  
  The SSL protocol allows connections to be secured using public-key encryption and authentication algorithms. “SSL support” on page 8 describes EAServer’s SSL support.
Network protocol support

To enable support for each protocol, you must define a listener in EAServer Manager. The listener configuration specifies a server address (host name and port number) as well as the network protocol and security settings to be used by clients that connect to that listener. SSL support requires installation of a server certificate. See the EAServer Security Administration and Programming Guide.

HTTP 1.1 support

EAServer supports HTTP/1.1, and complies with the features required for origin server and client in the W3C spec for HTTP/1.1. See the HTTP/1.1 specification at http://www.w3.org/Protocols/rfc2616/rfc2616.html.

HTTP tunnelling support

Almost all network firewalls allow HTTP traffic to pass, but some reject IIOP packets. When IIOP traffic is tunneled inside of HTTP, your clients can connect to EAServer through a firewall that does not allow IIOP traffic to pass.

The EAServer Java client ORB performs HTTP tunnelling automatically using the designated IIOP port. No additional configuration or proxies are required. When connecting, the EAServer client-side ORB first tries to open an IIOP connection to the specified address and port. If the IIOP connection fails, the ORB tries an HTTP-tunneled connection to the same address and port. The default behavior is appropriate when some users connect through firewalls that require tunnelling and others do not; the same application can serve both types. If you know HTTP tunnelling is always required for a Java client, you can set the ORBHttp property to cause the ORB to use HTTP tunnelling without trying plain IIOP connections first.

The C++ client ORB supports tunnelling when clients explicitly request it by setting the ORBHttp property.

SSL support

The SSL protocol allows connections to be secured using public-key encryption and authentication algorithms that are based on digital certificates. SSL is a wrapper protocol: packets for another protocol are secured by embedding them inside of SSL packets. For example, HTTPS is HTTP-secured by embedding each HTTP packet within an SSL packet. Likewise, IIOPS is IIOP embedded within SSL. HTTPS and IIOPS are also commonly called secure HTTP and secure IIOP, respectively.

EAServer provides native SSL protocol support. Specifically, the EAServer built-in SSL driver supports dynamic negotiation, cached and shared sessions, and authorization for client and server using X.509 Digital Certificate Support.

In EAServer Manager, you can configure a secure IIOP or HTTP port by defining an IIOP or HTTP listener, then associating a security profile with the listener. The security profile designates a server certificate that is sent to clients to verify that the connection ends at the intended server. The security profile also specifies the connection’s required security settings, such as:
• Whether a client certificate is required to open connections. The client certificate serves as proof of the client user’s identity.
• What data security options, such as the encryption algorithm, are used to secure data transmitted over the connection.

For detailed instructions on configuring secure ports, see the EAServer Security Administration and Programming Guide.

On the client-side, the following types of clients can open SSL connections to EAServer:
• Java applets hosted by SSL-capable Web browsers.
• Java applications
• C++ clients
• PowerBuilder clients
• ActiveX clients

Dynamic HTML support

You can use Java servlets, JavaServer Pages (JSPs), or PowerDynamo Web sites to dynamically create HTML pages and interactive HTML forms. Both PowerDynamo and Java servlets can call methods in EAServer components.

Java servlets are Java classes that use the standard javax.servlet API to respond to HTTP requests. EAServer can host Java servlets natively, and you can use EAServer Manager to associate logical paths in a HTTP URL with a Java servlet. JavaServer Pages extend the HTML page description language, allowing you to embed Java scriptlets within HTML tags. See these chapters in the EAServer Programmer’s Guide for more information:
• Chapter 22, “Creating Java Servlets”
• Chapter 24, “Creating JavaServer Pages”

If you have the PowerBuilder IDE, you can deploy JSP-based Web applications from PowerBuilder to EAServer. See Working with Web and JSP Targets in your PowerBuilder documentation set.

**PowerDynamo Web site conversion utility**

EAServer includes the Dyn2JSP utility to convert PowerDynamo Web sites into JSP-based J2EE Web applications. Sybase recommends that you migrate your PowerDynamo Web sites to the J2EE model. Support for PowerDynamo will be removed from later versions of EAServer. For information on using the Dyn2JSP utility, see the HTML documentation included in the PDynamo2JSP directory of your EAServer installation.

**Web server redirector plug-in**

EAServer hosts Web applications and functions as a Web server. You can install a redirector plug-in on the Web server host that allows you to send client requests directly to the Web server. The redirector plug-in enables communication between the Web server and the EAServer HTTP protocol listener. The redirector plug-in forwards requests to EAServer that need to access servlets, JSPs, and so on. EAServer processes the requests and returns the results back to the Web server.

See the EAServer Installation Guide for a list of the Web servers for which EAServer provides redirector plug-ins.

**Administration and development tools**

Sybase Central is a common management framework for Sybase application and database servers. EAServer provides two Sybase Central plug-ins for use by developers and administrators:
EAServer Manager provides graphical administration facilities for
EAServer, including support for development, deployment, runtime
monitoring of applications, and management of the server’s SSL digital
certificate database.

Standalone Security Manager can be installed on client machines where a
full EAServer installation is not required and provides a graphical user
interface for managing SSL digital certificates, which can be used by end
users to manage the certificates used in client applications.

For detailed instructions on running EAServer Manager and the standalone
Security Manager, see the EAServer System Administration Guide and the
EAServer Security Administration and Programming Guide.

Development support
Sybase PowerBuilder has been integrated with EAServer. Using this tool, you
can develop, deploy, and debug EAServer components entirely within the
development environment. You can also generate the proxies required for
client application development. For more information, see the Application
Techniques manual in the PowerBuilder documentation.

Application developers using other tools can use EAServer Manager to view
the method definitions for any installed component in EAServer Manager. You
can view and edit method definitions graphically, or you can directly edit the
IDL datatype and interface definitions with the EAServer Manager IDL editor.
You can import interface definitions from existing Java classes, ActiveX
component type library files, or from standard CORBA IDL files. See
“Defining Component Interfaces” in the EAServer Programmer’s Guide for
more information.

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

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

- **Clusters**  A cluster represents a set of servers that share configuration
  information and run the same set of components. For applications with
  thousands of clients, clusters provide support for load balancing and high
  availability. EAServer Manager includes a synchronization feature, which
  replicates component files and configuration information from a primary
  server to other servers in a cluster.

- **Servers**  A server represents one EAServer runtime process. Each server
  has its own network addresses for client session connections and for HTTP
  (HTML) connections. All servers on one host machine share the same
  configuration repository. For administration purposes, you can connect to
  any server on the host machine to configure other servers on the same host.
Applications allow you to group packages (groups of related components) and Web applications (bundled static Web content, servlets, and JSPs) into a single unit for easy deployment between servers.

Packages A package organizes components into cohesive, secure units that can be easily deployed on another EAServer installation. Packages can be exported, or saved, as a Java archive (JAR) file. The package archive includes the definition of all components in a package, plus any supporting files (such as source code and client files) that you specify. Package archives exported from one server can easily be imported for redeployment on another server.

Roles attached to packages control access to components in a package. For more information about roles and package security, see the EAServer Security Administration and Programming Guide.

Components A component definition consists of the component’s method signatures and other properties, such as component type, transaction support, threading model, and the name of the Java class or executable library that implements the component.

Web applications Web applications allow you to group static Web content, servlets, and JSPs into a single unit for easy deployment and configuration.

Web components You can install servlets and JSPs in a Web application as Web components. This allows you to configure request path mappings and other useful settings within the Web application properties.

Before a client application can execute a component, the component must be installed in an EAServer package, and that package must be installed in the server to which the client connects.

Hot refresh support EAServer Manager includes an option that allows you to refresh components, packages, and servers, enabling you to test and debug component implementation changes without restarting the server.

Performance monitoring support EAServer includes a Performance Monitor feature to prevent degradation of server performance under extreme load conditions. You can configure the Performance Monitor settings to heuristically govern the processing requests to prevent performance degradation due to overuse of available resources. For details, see Chapter 9, “Using the Performance Monitor,” in the EAServer Performance and Tuning Guide.

Runtime monitoring support The EAServer Manager plug-in for Sybase Central allows you to remotely view server log files and to monitor statistics for component execution and network activity.
Use the EAServer Manager File Viewer to connect to EAServer and view the contents of these server log files:

- The EAServer log file, where the server records errors related to component execution
- The HTTP protocol request and error log files, where EAServer’s built-in Web server records successful and unsuccessful file requests

Runtime Monitoring allows you to view statistics on component and network activity. You can view counts of active client sessions, components, and transactions.

See the *EAServer System Administration Guide* for more information on these features.

EAServer Manager | Certificates folder (server) or the standalone Security Manager (client) allows you to manage the server and user certificates that are required for SSL-protocol support. This allows you to:

- **Install server certificates** Server certificates are required to establish secure IIOP and HTTP ports. The certificate is presented to the client application as proof that the application has connected to the server that the user intends to interact with.

- **Install Certificate Administrator (CA) certificates** CA certificates, also called signing certificates, are attached to client and server certificates to validate the origin of the certificate. For example, if you obtain a certificate from VeriSign, your certificate includes a copy of the VeriSign CA certificate. Using the Certificates folder, you can install CA certificates into EAServer and indicate which CAs are trusted. When client applications present certificates to EAServer, the signing certificate must be present in the list of trusted CAs or the connection fails.

- **Issue certificates for testing purposes** EAServer Manager | Certificates folder allows you to create new client and server certificates for use in testing your applications. To deploy Internet applications, you must obtain server certificates from a well-known Certificate Authority (CA) such as VeriSign or Thawte. Certificates signed by the internal Sybase Jaguar User Test CA will not be recognized by client browsers unless the end user has installed the Sybase Test CA in their browser’s database of trusted CAs.

See the *EAServer Security Administration and Programming Guide* for more information on SSL certificates.
jagtool and jagant

jagtool is a command line interface that allows you to automate some EAServer development and deployment tasks. jagant lets you run jagtool commands from Jakarta Ant build files.

Ant is similar to make, but is platform-independent, and allows you to incorporate jagtool commands into build files. This powerful feature allows you to write build files that automate many development and deployment tasks. See Chapter 12, “Using jagtool and jagant,” in the EAServer System Administration Guide.

Repository versioning support

The EAServer configuration repository stores configuration and implementation files for installed application entities such as components, Web applications, JSPs, and Java servlets. Repository versioning allows you to save numbered versions of an object. Each version archive contains configuration properties and implementation files associated with the object. For example, before undertaking a new development phase, you might save a new major version of your J2EE application.

Client-session and component-lifecycle management

EAServer client sessions are established internally by the client stubs and proxies that applications use to invoke EAServer component methods. A component’s lifecycle determines how instances are allocated, bound to client sessions, and destroyed. EAServer manages both client sessions and component lifecycles without requiring any specialized knowledge on the part of the application developer.

Client applications use a stub or proxy object to invoke methods on EAServer. Internally, the stub or proxy object establishes a network connection between EAServer and a remote client. All the stub/proxy models discussed in “Client stub and proxy support” on page 33 require user-authentication parameters to instantiate a stub or proxy object. The communication protocol is also determined when the stub or proxy object is instantiated. Once the stub or proxy object exists, all details of network communication are hidden from the application developer.

The user name determines which components a client session can access. If a component does not allow access to the user, attempts to instantiate a proxy or stub fail. The EAServer Security Administration and Programming Guide describes security support in detail.
CHAPTER 1  Introducing EAServer

Component-lifecycle management

All stubs and proxies use IIOP to communicate with EAServer. MASP clients use the TDS protocol. “Network protocol support” on page 7 discusses client protocols in detail.

A component’s lifecycle determines how instances of a component are allocated, bound to specific client sessions, and destroyed. In the simplest case, an instance is allocated for each stub or proxy created by the client and is destroyed when the client explicitly requests destruction or when it disconnects, whichever happens first.

More sophisticated components can be coded to support instance pooling. Instance pooling allows EAServer to maintain a cache of component instances and bind them to client sessions on an as-needed basis. Instance pooling requires the following changes to your component:

- The component must provide activate and deactivate methods. EAServer calls the activate method just before an instance is bound to a client session. activate must be able to reset the component to an as-allocated state. EAServer calls deactivate just before an instance is unbound from a client session (that is, made idle again).

- Methods in the component must use the EAServer transaction state primitives to request early deactivation.

For components that support EAServer transactions, the time between EAServer activate and deactivate calls coincides with the beginning and end of that instance’s participation in an EAServer transaction.

Using components that support instance pooling increases the scalability of your application. Instance pooling eliminates execution time and memory consumption that would otherwise be spent allocating unnecessary component instances.


Load balancing and failover

In the simplest scenario, your application’s components are deployed to one server, and all clients connect to that server to execute the component’s business logic. This scenario works well when the number of simultaneous clients is not too large.

For applications with thousands of clients, you can define an EAServer cluster with several redundant servers to run the application’s components. The cluster allows load balancing and failover as follows:
Naming services

- **Load balancing**  EAServer can use load balancing, which optimizes performance for your EAServer cluster by adjusting the load across the servers, based on load metrics and a distribution policy, or based on a random algorithm. When a client resolves the name of an EAServer component, the name server returns several candidate server addresses. When you allow the algorithm to randomly distribute the processing load over servers in the cluster, the client ORB tries the addresses in random order. When you specify the load metrics and distribution policy, the load is distributed according to each server’s current load.

- **Failover**  You can configure components to allow automatic failover that is transparent to client applications. If a component allows automatic failover, the client ORB automatically reconnects to another server within the cluster when a previously connected server goes offline.

  In-memory failover support allows component state to be maintained on a pair of servers, without incurring the overhead of using a remote database to store component state. See Chapter 28, “Configuring Persistence for Stateful Session Components,” in the *EAServer Programmer’s Guide*.


Naming services

When multiple servers are involved in your application, the naming service allows you to specify logical server names rather than server addresses. For example, instead of connecting to your finance component server at host badger using port 9000, you can specify the initial naming context for that server, such as *USA/MyCompany/FinanceServer*. Components are identified by specifying an initial server name context plus the package and component name. For example:

*USA/MyCompany/FinanceServer/FinancePackage/PayrollAdmin*

This layer of abstraction allows you to move a server to another host without affecting deployed client applications. Naming does require that one EAServer installation use a well-known, stable host and port. This server acts as the name server for other servers that participate in your application, and clients connect to that server to resolve name requests.
You can use either persistent or transient storage for the naming database. For transient storage, EAServer builds the name database in memory when it starts, based on the contents of the EAServer configuration repository. For persistent storage, you must provide a third-party directory server that accepts connections using the Lightweight Directory Access Protocol (LDAP). When using persistent storage, EAServer connects to the third-party directory server to create and edit name database entries, and to resolve client name requests.

The naming configuration for a multiserver application is briefly summarized as follows:

1. Choose one server to act as name server for the application. You can configure this server to store names in memory (transient storage), or to store names in a third-party directory server (persistent storage).

2. Configure each of the remaining servers to connect to the designated EAServer naming server to resolve names. Each server also updates the name space when packages and components are added or deleted in EAServer Manager.

For detailed instructions, see Chapter 5, “Naming Services,” in the EAServer System Administration Guide.

Client APIs for naming

For Java and C++ clients, EAServer provides industry-standard client-side APIs for naming services.


PowerBuilder clients use the naming service implicitly. The name resolution is performed automatically when you create EAServer component instances using the CreateInstance and Lookup functions of the Connection object. You can browse the naming service using the CosNaming API, but such complexity is not necessary. See the Application Techniques manual in the PowerBuilder documentation.

ActiveX clients use the naming service implicitly.
Connection caching

Connection caching allows EAServer components to share pools of preallocated connections to a remote database server, avoiding the overhead imposed when each instance of a component creates a separate connection. Components that support transactions must use a connection from an EAServer connection cache to interact with remote databases.


The J2EE connector architecture enables you to write portable Java applications that can access multiple transactional enterprise information systems. A connector is a specialized connection factory that provides connections for EJBs, Java servlets, JSPs, and CORBA-Java components.

Transaction management

The EAServer transaction management feature allows you to specify a transaction coordinator—Java Transaction Service (JTS) or the Microsoft Distributed Transaction Coordinator (DTC)—and define a component’s transactional semantics as part of the component interface. See Chapter 2, “Understanding Transactions and Component Lifecycles,” in the EAServer Programmer’s Guide.

Localization

EAServer supports localization in the following ways:
**CHAPTER 1  Introducing EAServer**

- **Coded character set conversions**  EAServer supports multiple coded character sets for clients and components. When a client and component use different coded character sets, EAServer automatically converts character data from one character set to another. For example, if the client uses the roman8 character set and the component uses iso_1, EAServer converts string parameters and return values automatically from roman8 to iso_1 when the client calls the component methods.

In accordance with the Java and ActiveX standards, Java components, Java clients, ActiveX components, and ActiveX clients all use 16-bit Unicode. Unicode contains mappings for all characters in all other known coded character sets.

For C and C++ components, you can specify code sets using EAServer Manager. The server Codeset property specifies the default for all C and C++ components. You can override the default for an individual component by setting the Codeset component property.

For C++ clients, you can specify a character set when initializing the EAServer client ORB by setting the ORBCodeSet property. See Chapter 15, “Creating CORBA C++ Clients,” in the EAServer Programmer’s Guide.

For MASP clients, the code set is specified as an Open Client or ODBC property before the client connection is opened. See Appendix A, “Executing Methods As Stored Procedures,” in the EAServer Programmer’s Guide.

- **Localizing Web applications**  EAServer supports the HTTP 1.1 internationalization features defined in the Java Servlet 2.3 specification. Using these features, you can develop servlets that respond in the language specified by the request header, or configure localized versions of Web site’s static pages.


- **Using non-ASCII text and the message service**  You can store, retrieve, and display non-ASCII text correctly when using the message service database by:
  - Specifying the Server Name for the message service connection cache using a jConnect™ for JDBC™ URL.
  - Changing the character set of the message service database to be consistent with the language you are using.
Result-set support

See Chapter 8, “Setting up the Message Service,” in the *EAServer System Administration Guide*.

- **Coded character set interoperability with PowerBuilder** The Sybase document Guidelines for Code Set Interoperability with PowerBuilder and EAServer at http://content.sybase.com/detail?id=1028793 includes information about the following topics:
  - Transmission code set
  - Client code set
  - Component code set
  - PowerBuilder code set
  - The euro character

**Result-set support**

EAServer methods can return tabular data to the calling client. This feature can be useful for the following reasons:

- **Use with data-aware controls** Some front-end tools provide objects that can automatically display a result set. For example, if you are using PowerBuilder, you can return results in a DataStore object from component methods and display the results using a DataWindow® control in the client. PowerBuilder DataWindow technology, available in both Web and PowerBuilder clients, allows you to display result sets and synchronize updates with a minimum of coding.

- **Efficiency** For tasks that require returning tabular data, using an EAServer result set is the most efficient alternative. Common uses of result sets include menu and picklist population. For example, in an online clothing catalog, you must list in-stock sizes for each item.

The EAServer result set allows data to be sent all at once (rather than requiring a get-next-row method and one client-server round trip per method). A large EAServer result set can be sent with less overhead than is required to encapsulate tabular data as an object and send a serialized version of the object to the client.
Each component model provides an interface that allows you to define result sets from scratch or to forward results from a remote database query directly to the client. See Chapter 25, “Sending Result Sets,” in the *EAServer Programmer’s Guide*.

For information on using the PowerBuilder DataWindow, see the *PowerBuilder Application Techniques* manual.

**Support for asynchronous messaging**

Chapter 31, “Using the Message Service,” in the *EAServer Programmer’s Guide* describes EAServer messaging support. The EAServer message service supports the Java Message Service (JMS) 1.0.2 specification, which addresses the demands of distributed systems in a coherent manner. JMS offers an API and a set of semantics that prescribe the interface and general behavior of a messaging service.

The message service allows you to publish or send messages to a queue, where they are stored until they can be delivered to the queue’s recipient, which is either a client or a component. The message service provides a pull-style mechanism for client notification and a push-style mechanism for component notification. Clients can check their queue for new messages, or spawn a thread to wait for a message’s arrival. Components can also check for new messages and they can register to be notified when messages arrive in their queue.

The message service provides transient and persistent message storage for message consumers and allows message producers to send messages to a particular message queue, or to publish messages with specific topics, available to all message queues.

The message service is implemented as an EAServer component with interfaces specified in standard CORBA IDL. Consequently, it can be used by all types of clients and components.
Asynchronous processing support

Most server processing is driven by client interaction. However, some tasks are best performed asynchronously; for example, maintaining copies of cached data retrieved from a slower source, updating search indexes on a regular schedule, or performing lengthy calculations. EAServer provides two vehicles to support asynchronous processing:

- Service components can be used for processing that must run continuously for the life of the server process, or must be performed once each time the server is started. Service components run in a thread that is started when the server starts. See Chapter 33, “Creating Service Components,” in the EAServer Programmer’s Guide.

- The EAServer thread manager allows you to run threads at any time. Threads started by the thread manager execute independently of the client or component that starts them, and can be configured to run once or periodically at regular intervals. See Chapter 32, “Using the Thread Manager,” in the EAServer Programmer’s Guide.

External processing

The EAServer component model allows you incorporate legacy business logic code into a component. However, if legacy code is unstable, it can cause the server to crash.

Stateless components of any type can run externally, with full access to server-side features such as cached connections. When you mark a component to run externally, EAServer runs it in a separate server process. You can identify which external server runs the component, and assign groups of related components to run in the same external server. EAServer starts the external server when required, and restarts the server if it stops responding.


Systems management support

EAServer includes enhanced support for remote systems management. The implementation is based on the Java Management Extensions (JMX) agent management framework. It provides the following enhancements:

- Allows you to create management beans (MBeans) that can be run in a JMX framework.
CHAPTER 1    Introducing EAServer

- Supports both SNMP and JMX management tools.
- Allows you to monitor servers, clusters, and key server subsystems such as the message service, the component dispatcher, and network listeners.


Support for legacy Open Server applications

You can easily convert existing applications that are coded for Sybase Open Server to run in EAServer. You can recompile and link your existing event handler code into a DLL or shared library, then install it into EAServer. After you define an Open Server listener, your existing clients can connect to the listener’s port. No client changes are required, aside from the possible requirement that you change the address to which the client connects.


IPV6 support

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

IPV6 support
CHAPTER 2

Developing an EAServer Application

This chapter explains some of the basic concepts and terminology associated with developing component-based EAServer applications in a three-tier environment.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introductory concepts</td>
<td>25</td>
</tr>
<tr>
<td>Developing an EAServer application</td>
<td>27</td>
</tr>
<tr>
<td>EAServer runtime environment</td>
<td>28</td>
</tr>
</tbody>
</table>

Introductory concepts

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

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

- A package is a collection of components that work together to provide a service or some aspect of your application’s business logic. A package defines a boundary of trust within which components can easily communicate. Each package acts as a unit of distribution, grouping together application resources for ease of deployment and management.

EAServer supports the following types of components:

- EJB
- Java
- CORBA C++
- PowerBuilder NVO
- ActiveX
- C

- **A stub** is a Java class or a C++ stub generated by EAServer Manager and acts as a **proxy object** for an EAServer component. A stub is compiled and linked with your Java applets or client application. A stub communicates with EAServer to instantiate and invoke a method on a component in the middle tier. Stubs make a remote EAServer component appear local to the client.

- **A skeleton** acts as the interface between the EAServer runtime environment and the user code that implements the method. Skeletons are compiled and linked with each of the components, and at runtime they enable EAServer to locate and invoke an appropriate method.

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

- A **Web application** is a collection of static HTML pages, Java servlets, and JavaServer Pages. You can develop Web applications to provide a browser-based user interface as an alternative to standalone clients or Java applet clients.

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

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

Java applets are downloaded to clients, which instantiate components on the server. Client applications are installed on client machines, from which they also instantiate components on the server.
An applet, standalone application, or Web application manages presentation and interaction with an end user. Middle-tier components, which run in EAServer, handle much of the application processing. Finally, the database stores, manages, and processes data.

If the client is an applet, users find and launch applications from traditional HTML pages. Instead of simply loading a static page, EAServer downloads an executable applet to the individual’s browser. If the client is an already-installed application, the user launches the application from his or her machine. Clients communicate directly with an application component running in the middle tier. Server components access data from one or more databases, apply business logic, and return results to the client applet for display.

When a proxy object is created on the client applet, it instantiates a corresponding component registered with EAServer. On the server side, a component is instantiated in response to a request from the proxy object running in the client environment. A method on a component is executed when it is invoked by a proxy object on the client applet.

Web applications can call EAServer components using the same proxies as used by standalone Java clients and applet clients.

Developing an EAServer application

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

❖ Creating and deploying an EAServer application

1. Use EAServer Manager to define packages, components, and methods. EAServer Manager generates:
   - Client-side stub files – contain interface information used by the client to invoke EAServer component methods.
   - Server-side skeleton files – provide the interface information of each component method.

2. Once you have generated the stubs and skeletons, write the user interface logic for the client model that you have chosen.
Develop the server-side components that link with the skeletons to form the business logic of your servlet. EAServer supports many of the integrated development environment (IDE) tools available today.

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

### EAServer runtime environment

A typical EAServer application has an applet or HTML page associated with it. Once you build and deploy such an application, it runs in the following fashion:

1. EAServer receives an HTTP request and downloads the requested HTML page or applet. Included with the applet are the Java stubs, which through a proxy, instantiate components and invoke the methods on those components.

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

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

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

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

6. EAServer returns the results from the database to the client.
7 The client indicates that it has completed the operation. EAServer destroys the component instance or returns it to a pool for future client instantiations. The client disconnects from EAServer.
EAServer runtime environment
CHAPTER 3

EAServer Components

This chapter discusses the component models that EAServer supports.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>31</td>
</tr>
<tr>
<td>Enterprise JavaBeans components</td>
<td>35</td>
</tr>
<tr>
<td>Java components</td>
<td>39</td>
</tr>
<tr>
<td>CORBA-C++ components</td>
<td>40</td>
</tr>
<tr>
<td>PowerBuilder components</td>
<td>40</td>
</tr>
<tr>
<td>ActiveX components</td>
<td>41</td>
</tr>
<tr>
<td>C components</td>
<td>41</td>
</tr>
</tbody>
</table>

Overview

EAServer components contain the methods that execute business logic and access data sources. The EAServer server-side component support and client-side stub or proxy support are independent. Any EAServer client can execute any type of component. A component of any model can execute components of another model using intercomponent calls without the use of additional gateway software.

EAServer provides support for several major component models, including:

- **Enterprise JavaBeans**  EAServer supports Java components that follow the Java Software Enterprise JavaBean (EJB) specification, versions 1.0, 1.1, and 2.0. An Enterprise JavaBean is a nonvisual, transactional component that is implemented in Java. For more information, see “Enterprise JavaBeans components” on page 35, and Chapter 7, “Creating Enterprise JavaBeans Components,” in the EAServer Programmer’s Guide.
A message-driven bean (MDB) is a type of EJB specifically designed as a Java Message Service consumer. Chapter 31, “Using the Message Service,” in the *EAServer Programmer’s Guide* describes how to create MDBs.

- **CORBA-Java** CORBA-Java components follow the CORBA component model and use standard CORBA interfaces for transaction management. Almost any Java class with nonvisual behavior can be adapted to run as an EAServer component. Chapter 11, “Creating CORBA Java Components,” in the *EAServer Programmer’s Guide* describes the EAServer Java component support in detail.

- **PowerBuilder NVO components** Using PowerBuilder, you can create nonvisual objects (NVOs) that run natively in EAServer as EAServer components. You can also create NVO proxies for EAServer components, then use the proxies in PowerBuilder client applications. See the *Application Techniques* manual in the PowerBuilder documentation.

- **CORBA-C++ components** EAServer C++ components are C++ classes that contain methods with similar prototypes to the EAServer component interface; the exact interface mapping complies with the CORBA specification for IDL/C++ language bindings.


- **ActiveX** You can install any nonvisual ActiveX component as an EAServer component (though you may need to define an “adaptor,” or wrapper class to handle methods that use unsupported parameter datatypes). EAServer uses the Component Object Model (COM) and ActiveX automation support to execute ActiveX component methods. Consequently, all EAServer ActiveX components must support COM’s automation interface (the IDispatch interface). Many application development tools, such as Microsoft Visual Basic, can be used to create ActiveX components that are compatible with EAServer.

  Chapter 19, “Creating ActiveX Components,” in the *EAServer Programmer’s Guide* describes EAServer’s ActiveX support in detail.

EAServer currently implements subsets of the Microsoft Transaction Server (MTS) ActiveX interface with the goal of providing full compatibility in the future.
CHAPTER 3  EAServer Components

- **C components**  EAServer provides a C “pseudo” component model that you can use to adapt existing C procedural applications as EAServer components. C components are dynamic link libraries (DLLs) or UNIX shared libraries that contain C or C++ methods and method skeletons. Method skeletons contain C functions that retrieve the RPC’s parameters and invoke C component methods.


- **C++ classes can be run as C++ components**  Developers using EAServer version 1.1 may have adapted C++ classes to run as EAServer C components. In version 2.0 and later, these classes can be run directly as C++ components.

Applications invoke an EAServer component using a stub or proxy object. The stub or proxy acts as a local surrogate for the remote component; it provides the same method signatures as the component and hides the details of remote server communication. Stubs and proxies are available for:

- **Java (CORBA and EJB)**  Any component can be invoked via a Java stub class. EAServer Manager generates source code for Java stubs. At runtime, your client program instantiates the stub. When you call methods on the stub class, the stub transparently invokes the component method on EAServer. You can also create Java servlets or JavaServer pages that run in EAServer and call components. Using this model, you can create “zero-install” applications. With EAServer built-in HTTP support, these applications have no client-machine installation requirements other than the presence of a Web browser. You can add additional interactive functionality to browser-based clients using Java applets that run in the browser.

EAServer supports three Java client models:

- **EJB**  Your program uses the EJB (javax.ejb) classes and EAServer EJB stubs to call EAServer component methods. This client model follows the EJB 2.0 Specification and is backward compatible with the EJB 1.1 and 1.0 specifications. Chapter 8, “Creating Enterprise JavaBeans Clients,” in the *EAServer Programmer’s Guide* describes how to implement EJB clients.
Overview

- **CORBA-Java**  Your program uses the EAServer CORBA-compliant Java ORB or any other CORBA-compliant Java ORB to instantiate stubs. Stub method signatures are mapped from the component’s interface definition, based on the CORBA specification of IDL-Java language bindings. Chapter 12, “Creating CORBA Java Clients,” in the *EAServer Programmer’s Guide* describes how to implement CORBA-Java clients.

- **PowerBuilder**  PowerBuilder allows you to generate NVOs that act as proxies for EAServer components. Using a proxy, you can call component methods as if they were implemented as local NVO methods. See *Application Techniques* in the PowerBuilder documentation.

- **C++ (CORBA)**  Your program uses the EAServer CORBA-compliant C++ ORB or any other CORBA-compliant C++ ORB to instantiate stubs. Stub method signatures are mapped from the component’s interface definition, based on the CORBA specification of IDL-C++ language bindings. Chapter 15, “Creating CORBA C++ Clients,” in the *EAServer Programmer’s Guide* describes how to implement C++ clients.

- **ActiveX**  Your program invokes EAServer components using the EAServer ActiveX proxy. The ActiveX proxy allows you to invoke EAServer components from ActiveX-enabled visual builder tools such as Microsoft Visual Basic. Used on the server, the ActiveX proxy allows you to invoke any EAServer component from an ActiveX component. EAServer Manager generates the type-library information that is required to register the component interface with your development tool. The ActiveX proxy uses the EAServer C++ client ORB to communicate with EAServer. Chapter 20, “Creating ActiveX Clients,” in the *EAServer Programmer’s Guide* describes how to use the ActiveX proxy.

- **Methods As Stored Procedures (MASP)**  EAServer provides a built-in interface that allows you to execute component methods as if they were stored procedures in an Adaptive Server Enterprise database. You can use this interface to call EAServer components from scripting and user-interface-builder tools that are database-aware but that do not support ActiveX. Appendix A, “Executing Methods As Stored Procedures,” in the *EAServer Programmer’s Guide* describes how to use this client interface.

---

**CORBA**

CORBA is a distributed component architecture defined by the Object Management Group (OMG). EAServer supports the CORBA IIOP.

For information on the CORBA architecture, see the specifications available at the OMG Web site at http://www.omg.org.
Enterprise JavaBeans components

The Enterprise JavaBeans (EJB) technology defines a model for the development and deployment of reusable Java server components, called EJB components.

An EJB component is a nonvisual server component with methods that typically provide business logic in distributed applications. A remote client, called an EJB client, can invoke these methods, which typically results in database updates. Since EAServer uses CORBA as the basis for the EJB component support, EJB components running in EAServer can be called by any other type of EAServer client or component, and even CORBA clients using ORBs from other vendors that are compatible with CORBA 2.3.

The EJB architecture looks like this:

- **EJB server** The EJB server contains the EJB container, which provides the services required by the EJB component. EAServer is an EJB server.

- **EJB client** An EJB client usually provides the user-interface logic on a client machine. The EJB client makes calls to remote EJB components on a server and needs to know how to find the EJB server and how to interact with the EJB components. An EJB component can act as an EJB client by calling methods in another EJB component.

An EJB client does not communicate directly with an EJB component. The container provides proxy objects that implement the components home and remote interfaces. The component’s remote interface defines the business methods that can be called by the client. The client calls the home interface methods to create and destroy proxies for the remote interface.
Enterprise JavaBeans components

**EJB container**  The EJB specification defines a container as the environment in which one or more EJB components execute. The container provides the infrastructure required to run distributed components, allowing client and component developers to focus on programming business logic, and not system-level code. In EAServer, the container encapsulates:

- The client runtime and generated stub classes, which allow clients to execute components on a remote server as if they were local objects.
- The naming service, which allows clients to instantiate components by name, and components to obtain resources such as database connections by name.
- The EAServer component dispatcher, which executes the component’s implementation class and provides services such as transaction management, database connection pooling, and instance lifecycle management.

**EJB component implementation**  The Java class that runs in the server implements the bean’s business logic. The class must implement the remote interface methods and additional methods for lifecycle management.

**EJB component types**

You can implement three types of EJB components, each for a different purpose:

- Stateful session beans
- Stateless session beans
- Entity beans

**Stateful session beans**

A stateful session bean manages complex processes or tasks that require the accumulation of data, such as adding items to a Web catalog’s shopping cart. Stateful session beans have the following characteristics:

- They manage tasks that require more than one method call to complete, but are relatively short-lived. For example, a session bean might manage the process of making an airline reservation.
They typically store session state information in class instance data, and do not survive server crashes unless they are run in a cluster that has persistent storage enabled for the component.

There is an affinity between each instance and one client from the time the client creates the instance until it is destroyed by the client or by the server in response to an expired instance timeout limit.

For example, if you create a session bean on a Web server that tracks a user’s path through the site, the session bean is destroyed when the user leaves the site or idles beyond a specified time.

Stateless session beans

A stateless session bean manages tasks that do not require the keeping of client session data between method calls. Stateless session beans have the following characteristics:

- Method invocations do not depend on data stored by previous method invocations.
- There is no affinity between a component instance and a particular client. Each call to a client’s proxy can invoke a different instance.
- From the client’s perspective, different instances of the same component are identical.

Unlike stateful session beans, stateless session beans can be pooled by the server, improving overall application performance.

Entity beans

An entity bean models a business concept that is a real-world object. For example, an entity bean might represent a scheduled airplane flight, a seat on the airplane, or a passenger’s frequent-flyer account. Entity beans have the following characteristics:

- Each instance represents a row in a persistent database relation, such as a table, view, or the results of a complex query.
- The bean has a primary key that corresponds to the database relation’s key, and is represented by a Java datatype or class.
EJB transaction attribute values

Each EJB component has a transaction attribute that determines how instances of the component participate in transactions. In EAServer, you set the transaction attribute in the Transaction tab of the Component Properties dialog box.

When you design an EJB component, you must decide how the bean will manage transaction demarcation: either programmatically in the business methods, or by the container, based on the value of the transaction attribute in the deployment descriptor.

A session bean can use either bean-managed transaction demarcation or container-managed transaction demarcation; you cannot create a session bean where some methods use container-managed demarcation and others use bean-managed demarcation. An entity bean must use container-managed transaction demarcation.

EJB container services

The EJB container provides services to EJB components. The services include transaction and persistence support.

Transaction support An EJB container must support transactions. EJB specifications provide an approach to transaction management called declarative transaction management. In declarative transaction management, you specify the type of transaction support required by your EJB component. When the bean is deployed, the container provides the necessary transaction support.

Persistence support An EJB container can provide support for persistence of EJB components. An EJB component is persistent if it is capable of saving and retrieving its state. A persistent EJB component saves its state to some type of persistent storage (usually a file or a database). With persistence, an EJB component does not have to be re-created with each use.

An EJB component can manage its own persistence (by means of the logic you provide in the bean) or delegate persistence services to the EJB container. Container-managed persistence means that the data appears as member data and the container performs all data retrieval and storage operations for the EJB component. See Chapter 27, “Creating Entity Components,” in the EAServer Programmer’s Guide.
Java components

EAServer can load and execute a Java class file as a component. The class can be a standalone class or part of a JavaBeans component that does not display any graphics or text, that is, a nonvisual JavaBeans component.

The definition of a Java component specifies the interfaces that the component implements as well as its other properties.

All component interfaces for EAServer components are defined in CORBA IDL modules that are stored in EAServer’s IDL repository. Chapter 5, “Defining Component Interfaces,” in the EAServer Programmer’s Guide describes how to define IDL interfaces.

Java component developers typically use one of the following to define the interface or interfaces that their component implements:

- **Implement a Java source file and import the methods from it**  As an alternative to IDL, you can define a Java class or interface, then use EAServer Manager to import the method definitions from the compiled Java byte code file. EAServer creates a new component definition and an IDL interface that matches the methods defined in the Java file. For more information on this feature, see “Importing interfaces from compiled Java files” in Chapter 5, “Defining Component Interfaces,” in the EAServer Programmer’s Guide.

- **Use existing interfaces from EAServer’s IDL repository**  In some cases, client and server component developers may have agreed upon an existing interface or several interfaces that a component must implement. In this case, it is up to the component developer to implement the specified interface. EAServer stores HTML documentation for all interfaces in the IDL repository in the html/ir subdirectory of your EAServer installation.

- **Define one or more new IDL interfaces**  If you are defining the interface yourself, you can use EAServer Manager’s IDL editor to create a new interface for the component. “Defining modules, interfaces, and types in IDL” in Chapter 5, “Defining Component Interfaces,” in the EAServer Programmer’s Guide describes how.
CORBA-C++ components

EAServer provides a CORBA-compatible C++ client-side interface. This allows you to create CORBA EAServer C++ applications. C++ components and clients are also interoperable with clients and components using other technologies.

The dynamic invocation interface (DII) is not supported.

Defining components

Use EAServer Manager to define basic information (such as the component name and methods) about a C++ component, and generate files that are required to write the component's class implementation and to compile the class into a dynamic link library (on Windows) or shared library (on UNIX).

Write your component as a C++ class; the generated files include a class implementation template in which you can write your method logic. In addition, EAServer supplies an application programming interface that contains classes and methods that you can use to perform EAServer-specific tasks. You can use the EAServer API to write code to handle errors, cache connections to third-tier database servers, return result sets, manage transactions, share data between instances of the same component, retrieve a client's SSL certificate information, and make intercomponent calls.

PowerBuilder components

Using PowerBuilder, you can create nonvisual objects (NVOs) that run natively in EAServer as EAServer components. You can also create NVO proxies for EAServer components, then use the proxies in PowerBuilder client applications. Inside EAServer, PowerBuilder components run in the PowerBuilder Virtual Machine (PBVM), which allows the EAServer component dispatcher to call the methods in your NVO component.

The PowerBuilder integrated development environment (IDE) includes wizards to create and deploy components, and generate proxies for use in PowerBuilder based client applications. For details, see the Application Techniques manual in the PowerBuilder documentation.

The PowerBuilder IDE runs on Windows platforms, but you can deploy PowerBuilder components to EAServer on any platform for which a compatible PBVM is available, including most UNIX platforms. For more information, see the EAServer Release Bulletin for your platform.
ActiveX components

ActiveX/COM is a Microsoft component technology. Many IDE tools such as Visual Basic allow you to create ActiveX components and write code to call methods in registered ActiveX components.

Any nonvisual ActiveX component can be installed as an EAServer component (though you may need to define an “adaptor,” or wrapper class to handle methods that use unsupported parameter datatypes). EAServer uses COM and ActiveX automation support to execute ActiveX component methods. Consequently, all EAServer ActiveX components must support COM’s automation interface (the IDispatch interface). Many application development tools can be used to create ActiveX components that are compatible with EAServer. Once installed in EAServer, ActiveX components can be called by clients of any type.

To support ActiveX clients, EAServer provides an ActiveX automation server that interacts with the server using the C++ CORBA ORB and standard CORBA IIOP. Because ActiveX clients use IIOP rather than the DCOM network protocol, they can call EAServer components of any type and interact with servers running on platforms that do not support ActiveX.

No client managed transactions This version of EAServer does not provide an ActiveX client interface to manage transactions. Consequently, ActiveX clients cannot call component methods that have the Mandatory transaction attribute.

C components

C components provide a quasi-object model for the execution of a group of related C functions. Unlike a C++ object, separate instances of a C component lack a private data space. However, you can implement create and destroy methods to associate data with an instance of a C component.
C components
A Web application allows you to deploy interrelated Web content, JavaServer Pages (JSPs), and Java servlets as a cohesive unit, and configure the Web server properties required by the servlets and JSPs.

This chapter presents an overview of Web applications. For detailed information on developing Web applications, see Chapter 21, “Creating Web Applications,” in the EAServer Programmer’s Guide.

### Contents of a Web application

A Web application is a unit of deployment for interrelated Web content, JavaServer Pages (JSPs), and Java servlets. The Web application contains static files, servlet and JSP implementation classes, and a deployment descriptor that describes how the files, servlets, and JSPs are configured on the host server. The deployment descriptor also allows you to configure application-specific HTTP properties, such as MIME types and per-file security constraints. To tie it all together, a Web application provides an abstract naming convention for the JNDI names of database connections and EJBs.

A Web application represents a subset of the files available on a Web server. Each Web application has a root request path that forms a prefix for URLs that access the JSPs, servlets, and static pages; for example, http://myhost/Finance. Each Web application also has a context root, which is a directory in the server’s file system where the Web application’s files are deployed. In EAServer, the context root for Web application wapp is this directory in your EAServer installation:

```
$JAGUAR/Repository/WebApplication/wapp
```
Contents of a Web application

Web applications contain several types of components.

Servlet files

Servlets are Java classes that create HTML pages with dynamic content and respond to requests from client applications that are implemented as HTML forms. Servlets also allow you to execute business logic from a Web browser or any other client application that connects using the Hypertext Transfer Protocol (HTTP). See Chapter 22, “Creating Java Servlets,’’ in the EAServer Programmer’s Guide.

Web clients invoke a Web application’s servlets by prepending the Web application’s root request path to an alias that is mapped to the servlet. For example, the following URL invokes a servlet mapped to the alias “Account” in the application with root request path “Finance”:

http://myhost/Finance/Account?type=add

Servlets that are not part of a Web application are implicitly assigned to the EASDefault Web application. You can configure the properties for EASDefault the same as you do for other Web applications. See “The EASDefault Web application” in Chapter 21, “Creating Web Applications,” of the EAServer Programmer’s Guide.

JSP files and tag libraries

JavaServer Pages (JSPs) allow you to embed snippets of Java code into HTML pages to create dynamic content. JSP tag libraries allow you to extend the standard HTML markup tags with custom tags backed by Java classes. See Chapter 24, “Creating JavaServer Pages,” in the EAServer Programmer’s Guide for more information on creating JSPs. If you have the PowerBuilder IDE, you can deploy JSP-based Web applications from PowerBuilder to EAServer. For more information, see Working with Web and JSP Targets in your PowerBuilder documentation.
Static files

You can include files that provide static content for the site in the Web site, including HTML, images, sounds, and so forth. You can also include Java applet files. You can configure the application’s deployment descriptor to specify security constraints for static files and any unique MIME types required by your content.

You must deploy static files to the following subdirectory in your EAServer installation directory:

```
Repository/WebApplication/web-app
```

Where `web-app` is the name of the Web application. You can include subdirectories, which are reflected in your application’s URL namespace.

If you import a Web archive (WAR) file, the importer expands the application’s static files to this location.

Java classes

A Web application’s Java classes include the implementation class for each servlet and JSP, and any server-side utility classes used by the servlets and JSPs.

EAServer uses a custom class loader to run a Web application’s servlets and classes referenced by servlet and JSP code. This feature allows hot refresh of servlets and JSPs. The custom class loader also allows each Web application to run with its own effective Java class path.

EAServer also supports class sharing among components and servlets. You can configure custom class lists for components, Web applications, J2EE applications, or a server process. The custom list allows you to support hot refresh of the implementation, and limit the number of copies of shared classes that are loaded into server memory. For example, if a Web application calls an EJB component, you can configure the component and the Web application to share instances of the component stub classes and common utility classes. See Chapter 30, “Configuring Custom Java Class Lists,” in the EAServer Programmer’s Guide.
Deployment descriptor

The application’s deployment descriptor catalogs the servlets, JSPs, and files contained in the application, as well as the properties of each. The descriptor must be formatted in XML, using the DTD specified in the *Java Servlet Specification Version 2.3*. You can create a descriptor using EAServer Manager or another J2EE-compliant development tool.
Using PowerDynamo with EAServer

This chapter describes how you can use PowerDynamo and EAServer together to serve up applications that require dynamic Web pages and access components (Java/Enterprise JavaBeans, ActiveX, and C/C++) that run in EAServer.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAServer and PowerDynamo architecture</td>
<td>48</td>
</tr>
<tr>
<td>Setting up PowerDynamo as a client</td>
<td>51</td>
</tr>
</tbody>
</table>

Do not use PowerDynamo for new application development  Sybase recommends that you do not use PowerDynamo for new application development. Instead, use the J2EE-standard Java servlet and JSP technologies. Support for PowerDynamo will be removed from later versions of EAServer.

Some UNIX versions of EAServer do not include PowerDynamo. See your release notes and install documentation for a list of products included with your EAServer software.

EAServer includes the Dyn2JSP utility to convert PowerDynamo Web sites into JSP-based J2EE Web applications. Sybase recommends that you migrate your PowerDynamo Web sites to the J2EE model. Support for PowerDynamo will be removed from later versions of EAServer. For information on using the Dyn2JSP utility, see the HTML documentation included in the PDynamo2JSP directory of the installation.

The remainder of this chapter provides information that may be useful for maintenance of existing PowerDynamo applications.

For more information, see the PowerDynamo documentation, which includes these books:

- User’s Guide
- Reference Manual
EAServer and PowerDynamo architecture

Figure 5-1 demonstrates one scenario in which a PowerDynamo client connects to EAServer to process a PowerDynamo script, which contains a Methods As Stored Procedures (MASP) call to an EAServer component.

Figure 5-1: EAServer and PowerDynamo

These steps correspond to the numbers in Figure 5-1:

1. A browser requests a Web page. The Web page is a PowerDynamo script that contains HTML, SQL queries, and a MASP call to an EAServer component in that order.
2. The Web server passes the request to PowerDynamo.
3 The script is retrieved from the Web site (not shown in Figure 5-1). Processing of the script begins. The HTML and SQL queries within the PowerDynamo script are processed.

4 An ODBC or Open Client connection is made to a database to retrieve the necessary information for the SQL query.

5 The data is returned to PowerDynamo. Processing of the script continues until the EAServer component call is encountered.

6 Because a MASP call is being made, an ODBC or Open Client connection is made to access the EAServer component and the called method.

7 The component method is executed. If the method requires database access, a connection is made to a database to retrieve the appropriate data.

8 The data is returned to EAServer through an ODBC, Open Client, or JDBC connection.

9 The results from the EAServer function are returned to PowerDynamo through an ODBC or Open Client connection.

10 PowerDynamo passes the results, in HTML format, back to the Web server.

11 The results are passed back to the client through an HTTP connection.

Notes

- EAServer and PowerDynamo can access several different databases; this diagram displays only one database.
- EAServer components can access data from many different types of databases; you do not need a Sybase database to store data.
- The client, PowerDynamo, and EAServer can all reside on either one machine or on separate machines.

A Web author can embed calls to EAServer components within a PowerDynamo script or template. The PowerDynamo script executes in the regular manner until it encounters the call to the EAServer component. At this point, a connection (through PowerDynamo) is made to EAServer, the appropriate method is executed, and the results are returned to PowerDynamo. PowerDynamo processes the information and returns HTML to the client.

**Calling EAServer components from PowerDynamo scripts**

You can call EAServer components from within a PowerDynamo script by using:
EAServer and PowerDynamo architecture

- The PowerDynamo SQL tag (must return a result set)
- The PowerDynamo COMPONENT tag (may return a result set)
- The connection.CreateQuery method (must return a result set)
- The connection.createComponent method (must return a result set)
- The java.createComponent method (no restriction on the return)
- An ActiveX object through the PowerDynamo CreateObject method (no restriction on the return)

The SQL tag, COMPONENT tag, and the connection.CreateQuery and connection.createComponent methods allow PowerDynamo to call EAServer components as though they were MASP. Each MASP invocation creates an instance of the component, invokes the method and then destroys the component instance. You can also use the COMPONENT tag to access ActiveX and Java objects—however, just as with MASP, each invocation creates an instance of the component, invokes the method, and then destroys the component instance. The PowerDynamo CreateObject method allows you to create an instance of an ActiveX object and access its methods and properties from within a script. The ActiveX object exists until it goes out of scope or until the ActiveX variable is assigned a new value. The PowerDynamo java.createComponent method allows you to create an instance of a Java object and access its methods and properties from within a script. The Java object exists until it goes out of scope.

Using MASP verses ActiveX and Java

Within an application, there are a few benefits to creating an instance of an ActiveX (using the CreateObject method) or Java object (using the java.createComponent method) against an EAServer component instead of making a MASP method call:

- ActiveX or Java objects exist until they go out of scope, whereas MASP calls create an instance of the component, execute it, and destroy it immediately.
- ActiveX or Java objects can give any type of return. MASP calls from PowerDynamo can return only a result set.
- Once ActiveX or Java objects are created, they are treated like normal DynaScript objects. MASP calls (except the connection.createComponent method) require that you enter a full string each time a call is made. For more information about DynaScript objects, see “Writing DynaScripts” in the PowerDynamo User’s Guide.
If a script requires repeated use of an EAServer component, the creation of an ActiveX object or Java stub is the preferred method of working with an EAServer component and its methods. ActiveX, however, is a Microsoft technology, which means that it is available only on Microsoft operating systems. Java, on the other hand, is platform-independent, which means that Java objects can run anywhere. For a detailed explanation of MASP, see the *EAServer Programmer's Guide*.

### Setting up PowerDynamo as a client

For PowerDynamo to access EAServer as a client, the machine on which PowerDynamo is installed must be set up properly. This setup depends on whether you are using ActiveX, Java, or MASP calls from the PowerDynamo client to the EAServer components.

**Setting up PowerDynamo to use ActiveX**

1. Ensure that EAServer Manager is installed on the PowerDynamo machine. If you do not install EAServer Manager on the PowerDynamo machine, you must copy and register the TLB/REG files for the package onto the PowerDynamo machine.
2. Install the ActiveX Proxy Automation Server on the PowerDynamo machine.
3. From EAServer Manager, connect to the server that contains the required components.
4. Generate TLB/REG files for the package, which generates ActiveX proxy objects that are used to create instances on the client and invoke methods on remote EAServer components.
5. Verify that the package has been registered.

**Setting up PowerDynamo to use Java**

1. Ensure that EAServer Manager is installed on the PowerDynamo machine.
2. From EAServer Manager, generate stubs for the desired package.
3. Ensure that the Java Code Base path is in your CLASSPATH.
4. Compile the generated Java files.
Setting up PowerDynamo as a client

5 Ensure that the Sun Java VM has been set. This is the only VM that works with EAServer.
6 Ensure that PowerDynamo is configured for Java support.
7 Set your default EAServer settings.

❖ Setting up PowerDynamo to use MASP

Setting up PowerDynamo as a client machine to use MASP calls requires only that a connection to EAServer exists. There are two types of connections that you can create:
1 System 11 ODBC data source connection
2 Open Client connection

For detailed information on setting up Java or ActiveX clients, or accessing EAServer MASP methods from a client, see the EAServer Programmer's Guide.
CHAPTER 6
Using Message Bridge for Java with EAServer

This chapter discusses Message Bridge and how you can use it with EAServer.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Bridge overview</td>
<td>53</td>
</tr>
<tr>
<td>Using Message Bridge</td>
<td>54</td>
</tr>
<tr>
<td>Message Bridge and EAServer architecture</td>
<td>54</td>
</tr>
</tbody>
</table>

**Message Bridge overview**

Message Bridge is a powerful tool you can use to build applications that generate and consume documents and messages. It generates Java classes that can be used as part of an EAServer component to assist in parsing and constructing XML documents that conform to a known schema. It includes:

- A GUI to access to all its features
- Importers that support different types of schemas
- Schema compilers to generate DataBeans
- DataBean runtime infrastructure
- Developer artifacts that support use of DataBeans

Message Bridge helps you build applications that make use of structured messages, such as XML documents or messages exchanged between enterprise systems or business partners through New Era of Network adapters, using EAServer components. Message Bridge improves your productivity by modeling the schema of a document or message as Java classes. When used in an EAServer application, these classes provide an intuitive way to access and manipulate message content in memory, and to read and write messages to and from the network.
Using Message Bridge

Message Bridge provides a schema compiler that binds a document or message schema into Java classes. Each class provides access to the data content of the corresponding schema component through accessor (get) and mutator (set) methods similar to those used in standard JavaBeans. Because these classes model the data content of a document or message instance, they are called DataBeans. In short, a DataBean is a Java binding of a particular schema.

Using Message Bridge

Using the Message Bridge GUI, you can import two types of schemas: DTDs and XML. Message Bridge converts the imported schemas into neutral representations that you can modify, enhance, and group into projects with other related schemas.

Then you can generate DataBeans for the individual schema definitions you select. These DataBean classes abstract the data contained in documents or messages. Each DataBean leverages shared runtime classes—the DataBean framework—to perform its functions: serializing and deserializing content from the data stream, validating content, and providing a read/write in-memory representation of message data.

During design, Message Bridge also generates artifacts to assist you in using DataBeans in your applications. These artifacts—XML DTDs, XML schemas, and HTML documentation for DataBeans—facilitate development in various ways. For example, the XML DTD and schema provide you with content model descriptions of each DataBean. By using these content models, you can use your own XML-based tools, easily modeling runtime systems based on XML data authoring, manipulation, and transmission. The HTML documentation provides Java developers with a detailed view of each particular DataBean’s content model.

Message Bridge and EAServer architecture

Figure 6-1 demonstrates Message Bridge creating a DataBean from a schema. A developer uses the DataBean to parse and construct the XML documents.
These steps correspond with the numbers in Figure 6-1.

Steps 1 through 3 are performed at design time. Steps 4 through 7 are performed at runtime.

1. Import a schema into Message Bridge and define your message.
2. Generate a DataBean.
3. Write an EJB component with a method that accepts an XML document and uses the DataBean for parsing it. Write a servlet that accepts an XML document.
4. EAServer accepts an HTTP request that contains an XML document.
5. EAServer invokes the servlet.
6. The servlet calls the EJB component method. The EJB component uses the DataBean to parse the XML document and perform business logic. The EJB component uses another DataBean to build an XML reply document and return the XML to the servlet.
7. The servlet returns the XML document in an HTTP response to the caller.
For more information

For more information see the *Message Bridge for Java™ User’s Guide.*
Also see the Message Bridge samples and tutorial in the *Messagebridge* subdirectory of your EAServer installation.
Index

A
ActiveX
  See also ActiveX clients
clients, character sets for  19
client-side support  34
server-side support  32
setting up PowerDynamo to use  51
versus MASP  50
addresses, configuring network  7
architecture
  EJB components  35
  Message Bridge  54
asynchronous processing
  using service components  22
  using the message service  21
  using the Thread Manager  22
authentication, support for mutual SSL  8

C
C language
  C components, introduction to  41
  implementing components in  33
C++
  client support  34
  clients, introduction to  40
  components  32
  components, development procedure for  40
  caches, connection  18
certificates, SSL
  managing in Security Manager  13
character sets
  conversions to and from  19
  interoperability with PowerBuilder  20
  message service database  19
  non-ASCII text  19
clients
  See also ActiveX clients, C++ clients, Java clients

creating a component instance  28
establishing a session  28
session management and  14
types of  33
code set. See character sets
component methods
  calling  49
  calling from PowerDynamo  47
  example of calling from PowerDynamo  48
components
  ActiveX  41
  and connection caches  28
  C  41
  client stubs and proxies for  33
  CORBA-C++  40
  creation and destruction of  15
definition of  25
  EJB  35
  executing methods on  27
  external processing  22
  instantiating  27
  introduction to  4
  Java  39
  lifecycle management  15
  models supported  25
  overview  4
  PowerBuilder  40
  refreshing after modifying  12
  reloading with EAServer Manager  12
  supported types  31
  transactional  28
types of  4, 31
concepts  25
connection caches, support for  18
connections, types of  52
coroutine root for a Web application  43
conventions vii
CORBA and C++ clients  40
creating
  ActiveX components  41
Index

an EAServer application 27
c components 41
C++ components 40
Java components 39
types of 36
using transactions in 38
entity bean, EJB component type 36
errors, viewing in log files 13
external processing of components 22

D
DataBeans, Message Bridge 54
deployment descriptors for Web applications 46
deployment of EAServer packages 11
developing an application 27
DTD schemas, importing using Message Bridge 54
Dyn2PSP, conversion utility for PowerDynamo 10, 47
dynamic HTML, EAServer support for 9
Dynamo. See PowerDynamo

E
EAServer
components 31
connecting to PowerDynamo 52
creating an application 27
deploying an application 27
developing an application 27
HTML support in 9
overview 1
PowerDynamo example 48
runtime environment 28
server runtime 3
services 3
using Message Bridge with 53
using with PowerDynamo 47
EAServer Manager
application objects managed in 11
developer use of 11
overview of 11
reloading components with 12
runtime monitoring in 12
use during debugging 12
viewing log files with 13
EJB components
introduction to 35
features
administration and development tools 10
asynchronous messaging support 21
asynchronous processing support 22
client-session and component-lifecycle management 14
cOMPONENT support 4
connection caching 18
dynamic HTML support 9
EAServer execution engine 3
IPV6 support 23
J2EE platform support 5
legacy Open Server application support 23
naming services 16
network protocol support 7
performance monitoring 12
result-set support 20
systems management support 22
transaction management 18
Web-server redirector plug-in 10

H
HTML files in Web applications 45
HTTP
request 28
support for 7
tunneling 8
HTTPS
definition of 8
support for 8

I
IDL and C++ clients 40
IIOP
  packets, tunneling  8
  support for  7
IIOPS
  definition of  8
  support for  8
importing schemas, Message Bridge  54
instantiating components  27
Internet Protocol Version 6. See IPV6 support
introductory concepts  25
IPV6 support  23

J
J2EE
  application support  5
  EJB component support  5
  platform support in EAServer  5
  Web application support  5
JAGUAR environment variable  43
Jaguar Manager. See EAServer Manager
Java
  See also Java clients; Java components
classes for Web applications  45
clients  33
clients, character sets for  19
components  31, 32
components, character sets for  19
creating components  39
setting up PowerDynamo to use  51
versus MASP  50
JBuilder plug-in  3
JMX  22
JSP
  adding to a Web application  44
  converting PowerDynamo to  10, 47

L
listeners, configuring  8
localization  18
localizing
  Web applications  19
log file, viewing with EAServer Manager  13

M
MASP
  setting up PowerDynamo for  52
  versus ActiveX  50
  versus Java  50
Message Bridge  53
  architecture  54
  DataBeans  54
documentation  56
importing schemas  54
overview  53
  using with EAServer  53
message service, overview of  21
monitoring, runtime  12
multitier application development overview  26
mutual SSL authentication  8

N
naming service
  PowerBuilder clients  17
naming services
  explanation of  16
  support for  16
network
  addresses  7
  protocols  7
non-ASCII text with the message service  19

O
ORB, Java client  8
ORBHttp property  8
overview
  EAServer application development  25
  multitier application development  26

P
package, EAServer
  definition  25
  refreshing after modifying  12
  uses of  12
Index

performance monitoring 12
plug-in for JBuilder 3
port numbers, configuring for servers 8
ports, configuring secure 8
PowerBuilder
   character set interoperability with EAServer 20
   clients and the naming service 17
   components 40
   deploying JSP-based Web application 9
   NVO components 32
   NVOs as proxies 34
   result sets 20
   uses for 11
PowerDynamo
   calling component methods 48, 49
   connecting to EAServer 52
   documentation 47
   EAServer example 48
   running in EAServer 9
   setting up ActiveX 51
   setting up as an EAServer client 51
   setting up Java 51
   setting up MASP 52
   using with EAServer 47
Web site conversion utility 10, 47
profiles, security 8
protocols
   HTTP 7, 8
   IIOP 7
   SSL 7
   supported 7
   TDS 7
proxy objects
   and stubs 26
   definition of 33
   purpose of 33

R
result sets 20
root request path, Web application 43
runtime
   environment, EAServer 28
   monitoring with EAServer Manager 12
   server engine 3

S
secure ports, configuring 8
Security Manager, definition of 11
security profile, use of 8
servers
   as managed in EAServer Manager 11
   configuring network addresses for 8
   monitoring performance 12
   overview of 3
   protocols supported by 7
   services provided by 3
service components, definition of 4
services provided by EAServer 3
servlets, running in Web applications 44
session bean, EJB component type 36
session management 14
session, definition of 26
setting up MASP access 52
skeleton, definition of 26
SNMP 22
SSL protocol
   authentication, mutual 8
   certificates, managing with Security Manager 13
   explanation of 8
   support for 7, 8
stub object, definition of 33
stubs and proxy objects 26
Sybase Central
   EAServer plug-ins for 10
   explanation of 10
   systems management 22

T
TDS protocol 7
terminology of component based applications 25
Thread Manager, overview of 22
transaction management 18
transactions, use in EJB components 38
tunnelling, HTTP 8
typographical conventions vii
Index

U
Unicode coded character set 19

W
Web applications
alternatives for 26
contents of 44
converting PowerDynamo Web sites to JSP-based 10, 47
definition of 43
deploying files in 45
deployment descriptor for 46
Java classes for 45
JSP files, adding to 44
localizing 19
overview 43
servlet files 44

X
XML schemas, importing using Message Bridge 54
Index