



Sun Java™ System

Application Server 7

Enterprise Edition Getting Started

Guide

2004Q2 Update 2

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819-1638

Copyright © 2005 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

THIS PRODUCT CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF SUN MICROSYSTEMS, INC. USE, DISCLOSURE OR REPRODUCTION IS PROHIBITED WITHOUT THE PRIOR EXPRESS WRITTEN PERMISSION OF SUN MICROSYSTEMS, INC.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

Use is subject to license terms. This distribution may include materials developed by third parties.

Sun, Sun Microsystems, the Sun logo, Java, Solaris, Sun™ ONE, Sun™ ONE Studio, iPlanet, J2EE, J2SE, Enterprise JavaBeans, EJB, JavaServer Pages, JSP, JDBC, JDK, JVM, Java Naming and Directory Interface, JavaMail, and the Java Coffee Cup logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc.

UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd.

This product is covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright © 2005 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse <http://www.sun.com/patents> et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats-Unis et dans les autres pays.

CE PRODUIT CONTIENT DES INFORMATIONS CONFIDENTIELLES ET DES SECRETS COMMERCIAUX DE SUN MICROSYSTEMS, INC. SON UTILISATION, SA DIVULGATION ET SA REPRODUCTION SONT INTERDITES SANS L'AUTORISATION EXPRESSE, ECRITE ET PREALABLE DE SUN MICROSYSTEMS, INC.

L'utilisation est soumise aux termes de la Licence. Cette distribution peut comprendre des composants développés par des tierces parties.

Sun, Sun Microsystems, le logo Sun, Java, Solaris, Sun™ ONE, Sun™ ONE Studio, iPlanet, J2EE, J2SE, Enterprise JavaBeans, EJB, JavaServer Pages, JSP, JDBC, JDK, JVM, Java Naming and Directory Interface, JavaMail, et le logo Java Coffee Cup sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Ce produit est soumis à la législation américaine en matière de contrôle des exportations et peut être soumis à la réglementation en vigueur dans d'autres pays dans le domaine des exportations et importations. Les utilisations, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers les pays sous embargo américain, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exhaustive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

Contents

About this Guide	7
What's In This Guide	7
Using the Documentation	8
How This Guide is Organized	10
Documentation Conventions	11
General Conventions	11
Conventions Referring to Directories	12
Contacting Sun	13
Give Us Feedback	13
Obtain Training	13
Contact Product Support	13
 Chapter 1 Introduction to Sun Java System Application Server 7, Enterprise Edition	15
About Sun Java System Application Server	15
About Application Server Instances	16
About Administrative Domains	18
About Enterprise Edition Features	18
Clustering	18
High Availability	19
Scalability	20
Load Balancing	21
Session Persistence	22
About Session Persistence Types	22
About Session Persistence Configuration	24
About Single Sign-on Session Information	25
About SFSB Checkpointing	26
High-Availability Database	26

Tools for Configuring and Administering the Application Server	27
Chapter 2 Clustering Scenarios	29
HTTP Clustering Scenario	29
RMI/IIOP Clustering Scenario	31
Server-side Configuration	31
Client-side Configuration	32
Stand-alone Client	33
ACC Client	34
Chapter 3 Enterprise Features Configuration Tutorial	35
Preparing to Use the Tutorials	35
Installation Requirements	36
Required Steps Before Using the Tutorials	36
Overview of Tutorial Steps	37
Starting the Server	39
Setting the PATH Variable	39
Running asadmin start-domain	40
Verifying Server Start-up	41
Verifying Admin Server Start-up	41
Accessing the Administration Interface	41
Viewing the Admin Server's Event Log	42
Verifying Start-up of Server Instances	43
Verifying Instances using asadmin	43
Verifying Instances by Accessing the HTTP Server	44
Creating the loadbalancer.xml File	45
Adding a Cluster to the loadbalancer.xml File	46
Configuring Load Balancing	49
Configuring the Health Checker	49
Enabling Load Balancer Monitoring	50
Other Load Balancer Properties	52
Dynamic Reconfiguration using the Reload Poll Interval	52
Response Timeout	53
HTTPS Routing	53
Sample loadbalancer.xml File	53
Chapter 4 Cluster JSP Sample Application Tutorial	55
Preparing to use the Cluster JSP Sample Application Tutorial	55
Deploying the Cluster JSP Sample Application to a Cluster	56
Input Files for the cladmin Command	57
The cladmin Command Syntax	58
Running cladmin deploy	58

The asadmin Commands Supported by the cladmin Command	59
Requirements and Limitations	60
Starting the Application Server Instances Using cladmin	60
Verifying Application Deployment	61
Monitoring the Sample in the Application Server	63
Using the Administration Interface to View Logs	63
Viewing Logs Using the tail Command	64
Application-Generated Messages in the Event Log	64
Application-Generated Messages in the Access Log	65
Troubleshooting Deployment to an Application Server Instance	65
Adding the Sample Application to the Cluster	66
Applying Configuration Changes and Restarting the Web Server	67
Running the Application	68
Verifying HTTP Load Balancing	73
Steps for Verifying Load Balancing	73
Troubleshooting the Load-Balancer Plug-in	74
Finding Errors in Your loadbalancer.xml File	75
Using the Health Checker	76
Verifying HTTP Session Persistence	77
Quiescing a Server Instance	78
 Chapter 5 Summary and Next Steps	 81
 Index	 83

About this Guide

This preface describes the contents of Sun Java™ System Application Server 7 *Getting Started Guide*.

This preface addresses the following topics:

- [What's In This Guide](#)
- [Using the Documentation](#)
- [How This Guide is Organized](#)
- [Documentation Conventions](#)
- [Contacting Sun](#)

What's In This Guide

This *Getting Started Guide* is intended for first-time users of the Sun Java System Application Server. It offers a brief, hands-on means of gaining familiarity with Sun Java System Application Server, Enterprise Edition features, with an emphasis on demonstrating the load balancing and HTTP session persistence features. Prior application server and development experience are not prerequisites for the exercises in this guide, though you must install the server using *Sun Java System Application Server Installation Guide* before starting.

The guide first introduces you to the Sun Java System Application Server, including a brief overview of the clustering, load balancing, and session persistence features. Next, it steps you through the configuration of an environment for running the sample application in the exercises. It shows you how to deploy a sample application to demonstrate and verify load balancing and session persistence. It contains a list of further areas of investigation, and references to where to find more information.

Using the Documentation

The Sun Java System Application Server Standard and Enterprise Edition manuals are available as online files in Portable Document Format (PDF) and Hypertext Markup Language (HTML).

The following table lists tasks and concepts described in the Sun Java System Application Server manuals. The manuals marked *(updated for 7 2004Q2)* have been updated for the Sun Java System Application Server Standard and Enterprise Edition 7 2004Q2 Update 2 release. The manuals not marked in this way have not been updated since the version 7 Enterprise Edition release.

Table 1 Sun Java System Application Server Documentation Roadmap

For information about	See the following
<i>(Updated for 7 2004Q2)</i> Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of supported hardware, operating system, JDK, and JDBC/RDBMS.	<i>Release Notes</i>
Sun Java System Application Server 7 overview, including the features available with each product edition.	<i>Product Overview</i>
Diagrams and descriptions of server architecture and the benefits of the Sun Java System Application Server architectural approach.	<i>Server Architecture</i>
<i>(Updated for 7 2004Q2)</i> How to get started with the Sun Java System Application Server product. Includes a sample application tutorial. There are two guides, one for Standard Edition and one for Enterprise Edition.	<i>Getting Started Guide</i>
<i>(Updated for 7 2004Q2)</i> Installing the Sun Java System Application Server Standard Edition and Enterprise Edition software and its components, such as sample applications and the Administration interface. For the Enterprise Edition software, instructions are provided for implementing the high-availability configuration.	<i>Installation Guide</i>
<i>(Updated for 7 2004Q2)</i> Evaluating your system needs and enterprise to ensure that you deploy Sun Java System Application Server in a manner that best suits your site. General issues and concerns that you must be aware of when deploying an application server are also discussed.	<i>System Deployment Guide</i>
Creating and implementing Java™ 2 Platform, Enterprise Edition (J2EE™ platform) applications intended to run on the Sun Java System Application Server that follow the open Java standards model for J2EE components such as servlets, Enterprise JavaBeans™ (EJBs™), and JavaServer Pages™ (JSPs™). Includes general information about application design, developer tools, security, assembly, deployment, debugging, and creating lifecycle modules. A comprehensive Sun Java System Application Server glossary is included.	<i>Developer's Guide</i>

Table 1 Sun Java System Application Server Documentation Roadmap (*Continued*)

For information about	See the following
Creating and implementing J2EE web applications that follow the Java™ Servlet and JavaServer Pages (JSP) specifications on the Sun Java System Application Server. Discusses web application programming concepts and tasks, and provides sample code, implementation tips, and reference material. Topics include results caching, JSP precompilation, session management, security, deployment, SHTML, and CGI.	<i>Developer's Guide to Web Applications</i>
(Updated for 7 2004Q2) Creating and implementing J2EE applications that follow the open Java standards model for enterprise beans on the Sun Java System Application Server. Discusses Enterprise JavaBeans (EJB) programming concepts and tasks, and provides sample code, implementation tips, and reference material. Topics include container-managed persistence, read-only beans, and the XML and DTD files associated with enterprise beans.	<i>Developer's Guide to Enterprise JavaBeans Technology</i>
(Updated for 7 2004Q2) Creating Application Client Container (ACC) clients that access J2EE applications on the Sun Java System Application Server.	<i>Developer's Guide to Clients</i>
Creating web services in the Sun Java System Application Server environment.	<i>Developer's Guide to Web Services</i>
(Java™ Database Connectivity (JDBC™), transaction, Java Naming and Directory Interface™ (JNDI), Java™ Message Service (JMS), and JavaMail™ APIs.	<i>Developer's Guide to J2EE Services and APIs</i>
Creating custom NSAPI plug-ins.	<i>Developer's Guide to NSAPI</i>
(Updated for 7 2004Q2) Information and instructions on the configuration, management, and deployment of the Sun Java System Application Server subsystems and components, from both the Administration interface and the command-line interface. Topics include cluster management, the high-availability database, load balancing, and session persistence. A comprehensive Sun Java System Application Server glossary is included.	<i>Administration Guide</i>
Editing Sun Java System Application Server configuration files, such as the <code>server.xml</code> file.	<i>Administrator's Configuration File Reference</i>
Configuring and administering security for the Sun Java System Application Server operational environment. Includes information on general security, certificates, and SSL/TLS encryption. HTTP server-based security is also addressed.	<i>Administrator's Guide to Security</i>
Configuring and administering service provider implementation for J2EE™ Connector Architecture (CA) connectors for the Sun Java System Application Server. Topics include the Administration Tool, Pooling Monitor, deploying a JCA connector, and sample connectors and sample applications.	<i>J2EE CA Service Provider Implementation Administrator's Guide</i>
Migrating your applications to the new Sun Java System Application Server programming model, specifically from iPlanet Application Server 6.x and Sun ONE Application Server 7.0. Includes a sample migration.	<i>Migrating and Redeploying Server Applications Guide</i>
(Updated for 7 2004Q2) How and why to tune your Sun Java System Application Server to improve performance.	<i>Performance Tuning Guide</i>

Table 1 Sun Java System Application Server Documentation Roadmap (*Continued*)

For information about	See the following
(Updated for 7 2004Q2) Information on solving Sun Java System Application Server problems.	<i>Troubleshooting Guide</i>
(Updated for 7 2004Q2) Information on solving Sun Java System Application Server error messages.	<i>Error Message Reference</i>
(Updated for 7 2004Q2) Utility commands available with the Sun Java System Application Server; written in manpage style.	<i>Utility Reference Manual</i>
Using the Sun™ Java System Message Queue 3.5 software.	The Sun Java System Message Queue documentation at: http://docs.sun.com/db?p=prod/s1.s1msgqu
For information about	See the following
(Updated for 7 2004Q2) Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of supported hardware, operating system, JDK, and JDBC/RDBMS.	<i>Release Notes</i>

How This Guide is Organized

This guide provides a Sun Java System Application Server overview of Enterprise Edition features for those new to Sun Java System Application Server.

The following chapters are included in this guide:

- [Chapter 1, “Introduction to Sun Java System Application Server 7, Enterprise Edition”](#) provides an overview of the product, including a brief introduction to clustering, load balancing, HTTP session persistence, and the high-availability database.
- [Chapter 2, “Clustering Scenarios”](#) contains procedures to configure HTTP and RMI/IIOP load balancing and failover.
- [Chapter 3, “Enterprise Features Configuration Tutorial”](#) contains a list of task to do before starting the tutorials, a summary of steps to configure your server to use Enterprise features, and a tutorial on configuring your server to use clusters and HTTP session failover.
- [Chapter 4, “Cluster JSP Sample Application Tutorial”](#) describes the Cluster JSP application, how to deploy it to a cluster, and how to run it. It describes how you can use this application to verify load balancing and HTTP session persistence.

- [Chapter 5, “Summary and Next Steps”](#) describes the tasks you’ve completed in this guide and the next steps to take in working with the product.

Finally, an [Index](#) is provided.

Documentation Conventions

This section describes the types of conventions used throughout this guide:

- [General Conventions](#)
- [Conventions Referring to Directories](#)

General Conventions

The following general conventions are used in this guide:

- **File and directory paths** are given in UNIX® format (with forward slashes separating directory names). For Windows versions, the directory paths are the same, except that backslashes are used to separate directories.
- **URLs** are given in the format:

<http://server.domain/path/file.html>

In these URLs, *server* is the server name where applications are run; *domain* is your Internet domain name; *path* is the server’s directory structure; and *file* is an individual filename. Italic items in URLs are placeholders.

- **Font conventions** include:
 - The monospace font is used for sample code and code listings, API and language elements (such as function names and class names), file names, pathnames, directory names, and HTML tags.
 - *Italic* type is used for code variables.
 - *Italic* type is also used for book titles, emphasis, variables and placeholders, and words used in the literal sense.
 - **Bold** type is used as either a paragraph lead-in or to indicate words used in the literal sense.

- **Installation root directories** for most platforms are indicated by *install_dir* in this document. Exceptions are noted in [“Conventions Referring to Directories” on page 12](#).

By default, the location of *install_dir* on **most** platforms is:

- Solaris and Linux file-based installations:

user's home directory/sun/appserver7

- Windows, all installations:

system drive: \Sun\AppServer7

For the platforms listed above, *default_config_dir* and *install_config_dir* are identical to *install_dir*. See [“Conventions Referring to Directories” on page 12](#) for exceptions and additional information.

- **Instance root directories** are indicated by *instance_dir* in this document, which is an abbreviation for the following:
default_config_dir/domains/domain/instance
- **UNIX-specific descriptions** throughout this manual apply to the Linux operating system as well, except where Linux is specifically mentioned.

Conventions Referring to Directories

By default, when using the Solaris package-based or Linux RPM-based installation, the application server files are spread across several root directories. This guide uses the following document conventions to correspond to the various default installation directories provided:

- *install_dir* refers to /opt/SUNWappserver7, which contains the static portion of the installation image. All utilities, executables, and libraries that make up the application server reside in this location.
- *default_config_dir* refers to /var/opt/SUNWappserver7/domains, which is the default location for any domains that are created.
- *install_config_dir* refers to /etc/opt/SUNWappserver7/config, which contains installation-wide configuration information such as licenses and the master list of administrative domains configured for this installation.

NOTE Forte for Java has been renamed to Sun Java Studio throughout this manual.

Contacting Sun

You might want to contact Sun Microsystems in order to:

- [Give Us Feedback](#)
- [Obtain Training](#)
- [Contact Product Support](#)

Give Us Feedback

If you have general feedback on the product or documentation, please send this to:

<http://www.sun.com/hwdocs/feedback/>

Obtain Training

Application Server training courses are available at:

http://training.sun.com/US/catalog/enterprise/web_application.html/

Visit this site often for new course availability on the Sun Java System Application Server.

Contact Product Support

If you have problems with your system, contact customer support using one of the following mechanisms:

- The online support web site at:
<http://www.sun.com/supporttraining/>
- The telephone dispatch number associated with your maintenance contract

Please have the following information available prior to contacting support. This helps to ensure that our support staff can best assist you in resolving problems:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem. Here are some of the commonly used commands:

- **Solaris:** `pkginfo, showrev`
- **Linux:** `rpm`
- **All:** `asadmin version --verbose`
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps
- Configuration files such as:
 - *instance_dir*/config/server.xml
 - a web application's `web.xml` file,
when a web application is involved in the problem
- For an application, whether the problem appears when it is running in a cluster or standalone

Introduction to Sun Java System Application Server 7, Enterprise Edition

The following topics are included in this chapter:

- [About Sun Java System Application Server](#)
- [About Enterprise Edition Features](#)
- [Tools for Configuring and Administering the Application Server](#)

About Sun Java System Application Server

The Sun Java™ System Application Server 7, Enterprise Edition provides a high-performance Java™ 2 Platform, Enterprise Edition (J2EE™) platform suitable for broad deployment of application services and web services. It offers a new modular architecture based on industry standard components, including proven implementations of the HTTP server infrastructure, Java Message Service (JMS), and rigorous support of the latest J2EE and web services specifications with J2EE version 1.3, Java 2 Platform, Standard Edition version 1.4, and the Java APIs for XML (JAX) from the Java Web Services Developer Pack software.

In addition, Sun Java System Application Server offers load balancing of HTTP/S and RMI/IIOP requests, and high availability of remote EJB references and HTTP sessions. For a complete list of features, see the *Sun Java System Application Server Product Introduction*.

This section describes the following topics:

- [About Application Server Instances](#)

- [About Administrative Domains](#)

About Application Server Instances

Application server instances form the basis of an application server deployment. Each instance has its own directory structure, configuration, and deployed applications. Each server instance also includes the J2EE platform web and EJB containers. The Sun Java System Application Server also includes a proven, high-performance HTTP server. An Object Request Broker (ORB) module enables EJB invocation using RMI-IIOP.

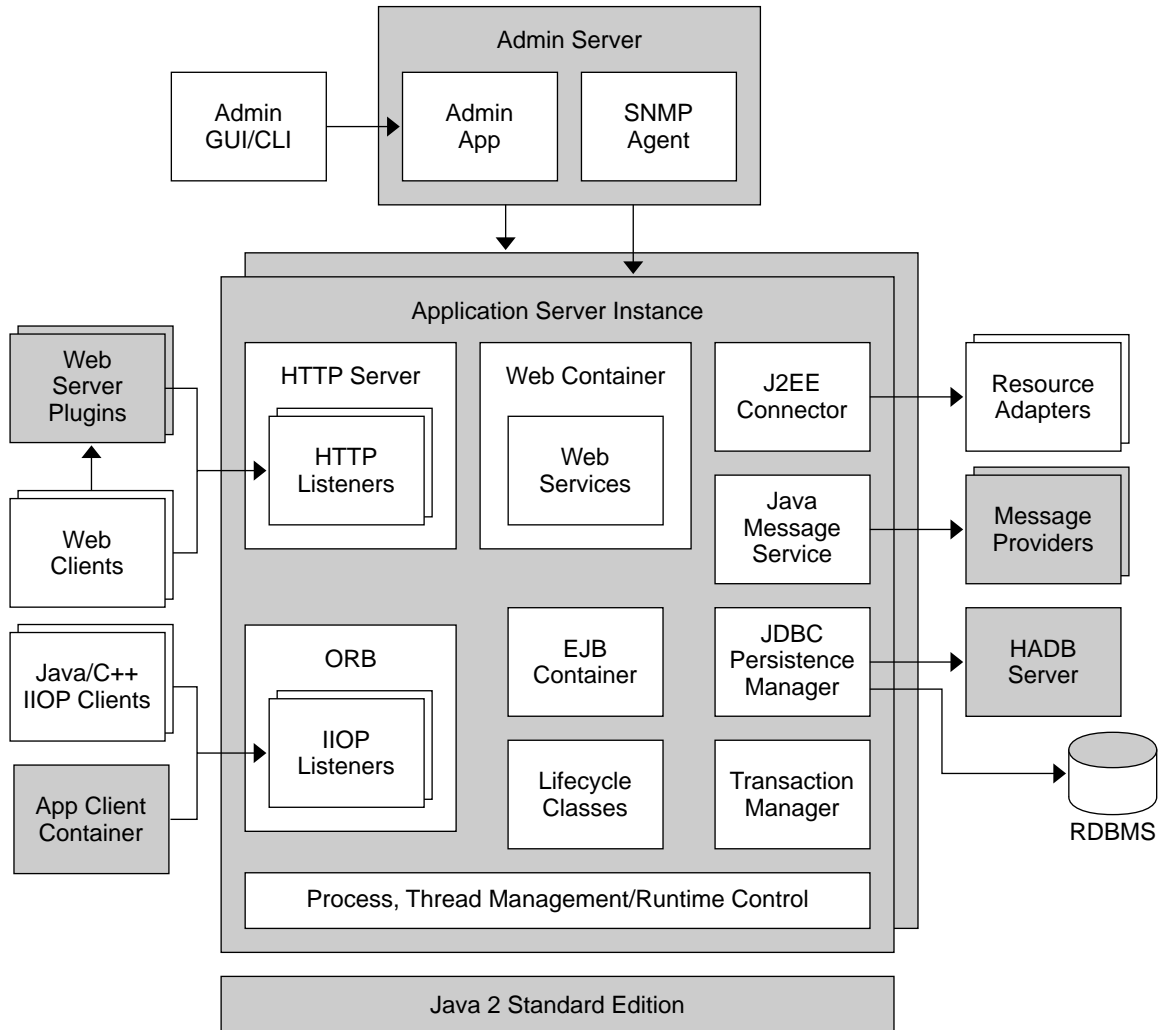
To access backend systems, applications can leverage J2EE connector architecture support and third-party resource adapters, either the built-in JMS provider or third-party providers, and any combination of popular third-party JDBC drivers. Access to backend systems can be managed within the scope of distributed transactions using the built-in, all-Java Transaction Manager.

The Admin Server (also called the Administration Server) is a special server instance that runs the core administration application and an SNMP agent. All remote management activity flows through the Admin Server. Both command-line and web-browser -based administrative clients access the Admin Server directly either through HTTP, or securely through HTTPS.

Web server plug-ins (such as the load balancer plug-in) enable you to deploy the Application Server behind one or more web servers housed in a demilitarized zone (DMZ) that is bracketed by one or more layers of firewalls. The plug-ins provide a means for the front-end web server tier to direct incoming HTTP and HTTPS traffic received from the Internet to one or more Application Servers housed in a back-end application server tier.

A variety of client applications can access business services deployed to the Application Server. Web service clients and browser-based clients can use either HTTP or HTTPS to access the web services, server-side end points, and J2EE web applications.

[Figure 1-1](#) shows an application server instance in detail. The application server instance is a building block in the clustering, load balancing, and session persistence features described in [“About Enterprise Edition Features” on page 18](#).

Figure 1-1 Sun Java System Application Server Instance

About Administrative Domains

Administrative domains enable you to define multiple, completely separate application server runtime configurations that reuse the same installation image. Each administrative domain has an Admin Server which in turn controls one or more application server instances. [Figure 1-1](#) shows a single administrative domain.

In the tutorials, you will be working with a single administrative domain configured during installation of the product, as well as multiple server instances in a cluster.

About Enterprise Edition Features

Sun Java System Application Server Enterprise Edition 7 2004Q2 provides advanced clustering and failover technologies. These features enable you to run scalable and highly available J2EE applications.

This guide shows you how to use the Enterprise Edition features described in the following sections:

- [Clustering](#)
- [Session Persistence](#)
- [High-Availability Database](#)

Clustering

A cluster is a group of application server instances that work together as one logical entity. Each Application Server instance in the cluster has the same configuration and the same applications deployed to it. For a detailed description of an Application Server cluster, see [Chapter 2, “Clustering Scenarios” on page 29](#).

The Application Server instances within a cluster can be hosted on different machines or on the same machine. That is, you can group application server instances across different machines into one logical cluster. For this guide, the default configuration is for two instances on the same machine.

For more information about clustering, see the *Sun Java System Application Server Administration Guide*.

Using clusters in Sun Java System Application Server helps you to achieve the following:

- [High Availability](#)
- [Scalability](#)
- [Load Balancing](#)

High Availability

Sun Java System Application Server provides high availability by allowing for failover protection of Application Server instances in a cluster. If one application server instance goes down, another Application Server instance will take over the sessions that were assigned to the unavailable server.

Session information is stored in the high-availability database (HADB) that is bundled with Sun Java System Application Server Enterprise Edition. For more information on the HADB, see [“High-Availability Database” on page 26](#).

Sun Java System Application Server provides failover support for the following:

- [HTTP/S Sessions](#)
- [EJB References Within HTTP Sessions](#)
- [Remote References of EJB and JNDI InitialContext over RMI/IIOP](#)
- [JMS connections](#)

HTTP/S Sessions

The load balancer plug-in fails over HTTP/S connections and the associated session information to another Application Server instance if the original Application Server instance serving the session becomes unavailable. Session information is stored in the HADB, when high availability is enabled.

The load balancing plug-in keeps track of HTTP/S sessions through the following two methods:

- Cookies
- Explicit URL rewriting

Cluster configuration should be created in the `loadbalancer.xml` file. This file has to be created manually, based on the `loadbalancer.xml.example` file that is installed along with the load balancer plug-in. By default, the example configuration file is located in your web server's `config` directory.

For more information on configuring high availability of HTTP/S sessions, see [Chapter 4, “Cluster JSP Sample Application Tutorial” on page 55](#).

EJB References Within HTTP Sessions

Failover of EJB references and J2EE objects that implement the `java.io.Serializable` interface that are stored in HTTP Sessions is supported. For more information, see “Configuring Session Persistence” in *Sun Java System Application Server Administration Guide*.

Remote References of EJB and JNDI InitialContext over RMI/IIOP

To enable high availability for RMI/IIOP applications, you need to configure an IIOP cluster with designated IIOP endpoints. The ORB *listeners* that will be part of the IIOP cluster, are called as IIOP *endpoints*. IIOP endpoints are configured using the Admin Console or CLI.

When a request is received along the RMI/IIOP path, one of the available IIOP endpoints in the cluster is selected randomly by the Application Server as the *primary* endpoint. The other IIOP endpoints in the cluster are then designated as *alternate* endpoints.

If the primary endpoint becomes unavailable, the remote reference associated with the connection is failed over to one of the alternate endpoints.

An IIOP cluster can be configured using either the Admin Console or CLI. The changes are registered in the Application Server configuration file, `server.xml`. For a detailed description, see [“RMI/IIOP Clustering Scenario” on page 31](#).

NOTE	The IIOP and HTTP clusters must be made up of the same Application Server instances.
-------------	--

JMS connections

JMS connection failover is provided by Sun Java System Message Queue Enterprise Edition 3.5 SP2.

Scalability

Sun Java System Application Server provides high scalability for your J2EE applications by allowing for the addition of Application Server instances to a cluster, thus increasing the capacity of the system. You can add Application Server instances to a cluster without disrupting service.

The HTTP and RMI/IIOP load balancing system distributes requests to healthy Application Server instances in the cluster.

Load Balancing

The goal of load balancing is to evenly distribute the workload among multiple Sun Java System Application Server instances. Load balancing can be configured for application requests along HTTP/S and RMI/IIOP path.

This section discusses the following topics:

- [HTTP Load Balancing](#)
- [RMI/IIOP Load Balancing](#)

HTTP Load Balancing

Sun Java System Application Server distributes incoming HTTP and HTTPS requests to Application Server instances that are configured in a cluster. Load balancing is accomplished by the bundled load balancer plug-in that is installed on a supported web server.

When a new HTTP request is sent to the load balancer plug-in, the request is forwarded to an Application Server instance based on a simple round robin scheme. Subsequent requests, with the same context-root, is assigned to the Application Server instance that initially serviced the request, based on cookies or explicit URL rewriting.

To configure HTTP load balancing and failover, you need to specify the Application Server cluster to the load balancing plug-in. The configuration changes must be made in the `loadbalancer.xml` file. For more information on the load balancer configuration file, see [“Creating the loadbalancer.xml File” on page 45](#).

You can either use the load balancer plug-in supplied with the Sun Java System Application Server, or you can use various third party hardware and software load balancers. This guide describes the bundled HTTP load balancer plug-in. For more information about load balancing, see the *Sun Java System Application Server Administration Guide*.

RMI/IIOP Load Balancing

Sun Java System Application Server, Enterprise Edition 7 provides load balancing of remote EJB references on the RMI/IIOP path, based on the JNDI InitialContext. A new target Application Server instance is selected, from among the configured IIOP cluster, for every new JNDI InitialContext initiated.

To enable load balancing for RMI/IIOP based requests, a few, minor code changes are required in the RMI/IIOP client application to enable load balancing. The bundled ORB provides the necessary infrastructure for load balancing.

Load balancing is supported for the following RMI/IIOP clients.

- Java applications executing in the Application Client Container (ACC) accessing EJBs deployed on an Application Server instance.
- Java applications, not running in the ACC, accessing EJBs deployed on an Application Server instance.

The settings for enabling load balancing in RMI/IIOP clients varies depending on the type of client. For an example configuration, see [“RMI/IIOP Clustering Scenario” on page 31](#). For detailed information on setting up various RMI/IIOP clients, see *Sun Java System Application Server Developer's Guide to Clients*.

Session Persistence

Session persistence ensures that if a Sun Java System Application Server instance or machine fails, the HTTP/S or EJB session will be picked up by another server instance. Sun Java System Application Server supports persistence of the following:

- HTTP sessions
- EJB references within HTTP sessions
- Stateful Session Beans (SFSB)

The high-availability database (HADB), bundled with the Sun Java System Application Server, functions as the persistence store. For more information on the high-availability database, see [“High-Availability Database” on page 26](#).

This section contains the following topics:

- [About Session Persistence Types](#)
- [About Session Persistence Configuration](#)
- [About Single Sign-on Session Information](#)
- [About SFSB Checkpointing](#)

About Session Persistence Types

Sun Java System Application Server supports three types of persistence: `ha`, `file`, and `memory`.

- The `ha` (high availability) persistence type stores the session information in the HADB. The `ha` persistence type is supported for production environments that require failover capabilities.

Use the Administration interface to choose the HADB persistence store. Follow these steps to set the HADB as your persistence store:

- Open the Availability Service component under your server instance.
- Go to the Availability Service page.
- Check the Instance Level Availability box.
- Click on the Save button.
- Click Properties under Persistence Store Properties.
- In the Name field, type `store-pool-jndi-name`.
- In the Value field, type the JNDI name of the HADB JDBC Resource. The assumed default is `jdbc/hastore`.
- Click on the Save button.
- Go to the server instance page.
- Apply Changes and restart the server.

The element hierarchy in the `server.xml` file looks like the following when the HADB is configured:

```
<server name="server1" ... >
    ...
    <availability-service availability-enabled="true">
        <persistence-store>
            <property name="store-pool-jndi-name" value="jdbc/hastore"/>
        </persistence-store>
    </availability-service>
    ...
</server>
```

For information about how to configure the HADB persistence store, see the Administration Guide.

- The `file` persistence type stores session information periodically to files, but you may experience loss of data when an instance becomes unavailable. The `file` persistence type is *not* supported for production environments that require failover capabilities; however it is sometimes used in development environments for testing applications.

The `session-store` attribute in the `server.xml` file determines where the SFSB state is stored if the local file system is used for SFSB state persistence. For example:

```
<server name="server1" ... session-store="/export/sfsbstore">
```

- The `memory` persistence type stores session information in a file in cases of graceful shutdown of the server instance. It does not store session information for unexpected shutdown. The `memory` persistence type is *not* supported for production environments that require failover capabilities.

The `ha` session persistence is the only type covered in the exercises in this guide. For more information on configuring and using the other types of session persistence, see the *Sun Java System Application Server Administration Guide*.

About Session Persistence Configuration

When you install the Sun Java System Application Server and run the `clsetup` command, your session persistence information is configured to default values. However, you can change the default values to balance your particular needs for performance, reliability, and high availability. These options are described in the following two topics:

- [Setting the Persistence Scope](#)
- [Setting the Persistence Frequency](#)

Setting the Persistence Scope

If you are using HADB as the persistence store, you must set the persistence scope to configure the amount of session state that should be stored. For example, every time your data is saved, or store the entire session, or store the session only if the session has been modified. You can also configure the persistence setting so that only the modified attributes of the session are stored.

The following three options are available:

modified-session The session is saved only if it has been modified.

session The entire session is saved every time session state is saved to the HADB.

modified-attribute Only the modified attributes of the session are saved.

Setting the Persistence Frequency

If you are using HADB to store session state, you can configure the frequency at which the session state is stored in the HADB database. For example, you can choose to store the session after every web request, thus providing high availability and reliability of updated session states, or you can choose to store the session after a time interval you specify, thus providing better performance.

The following two options are available:

web-method In the `web-method` persistence frequency, the session is stored at the end of every web request.

time-based In the `time-based` persistence frequency, the session state is stored at time intervals defined in the `reapIntervalSeconds` property of the `manager-properties` element in the `server.xml` file (for instance-level configuration) or in the `sun-web.xml` file (for application-level configuration).

NOTE You can choose the persistence frequency for only the `ha` persistence type.

For more information about the various configuration options for session persistence, see “Configuring Session Persistence,” in *Sun Java System Application Server Administration Guide*.

About Single Sign-on Session Information

In a single application server instance, once you are authenticated by a web application, you are not required to reauthenticate individually to other web applications running on the same instance. This is called single sign-on.

In order for this feature to continue to work even when a session fails over to another instance in a cluster, single sign-on information must be persisted to the HADB. This persistence is enabled when you enable high availability for the Application Server instance and the following property is added to the `virtual-server` element in `server.xml`:

```
<property name="sso-enabled" value="true"/>
```

High-availability is automatically configured when you run the `clsetup` command.

About SFSB Checkpointing

The state of an SFSB is saved to the persistent store at predefined points in its life cycle.

SFSB checkpointing can be enabled at five different levels:

- The server instance
- The EJB container
- The application
- The EJB module
- The SFSB itself

For SFSB checkpointing to be enabled at a given level, it must be enabled at all higher levels as well. For example, to enable SFSB checkpointing at the application level, you must also enable it at the server instance and EJB container levels.

For more information on how to configure SFSB checkpointing, see *Sun Java System Application Server 7 Developer's Guide to Enterprise JavaBeans Technology*.

High-Availability Database

The high-availability database (HADB) is used to store HTTP and SFSB session information. High availability means that the system remains available despite unplanned outages caused by hardware or software failures. The HADB, a JDBC-compliant database, is based on the Always-On technology and is capable of exceeding 99.999% data availability, offering an ideal platform for delivering all types of session state persistence within a highly-loaded enterprise application server environment.

The HADB achieves this data availability through fragmentation and replication of data. All tables in the database are partitioned to create subsets of approximately the same size called fragments. This process of fragmentation is based on a hash function that fragments and evenly distributes the data among the database's nodes. Each fragment is stored twice in the database, in mirror nodes. This ensures fault tolerance and fast recovery of data. In addition, if a node fails or is shut down, a spare node can take over until the node is active again.

The HADB stores and retrieves all state information in a separate but tightly-integrated persistent storage tier. For more information on setting up and configuring the HADB, see "Preparing for HADB Setup" in *Sun Java System Application Server Installation Guide*.

Tools for Configuring and Administering the Application Server

Sun Java System Application Server provides the following tools for configuring and administering the Sun Java System Application Server. You can use these tools to start and stop the server, as well as perform various other functions. Detailed steps for using some of these tools to perform specific configuration tasks are included in the tutorials.

- **The `asadmin` utility.** The `asadmin` utility is a command-line interface that you can use to perform administrative tasks on a single application server instance or administrative domain.

For more information on `asadmin`, see “Appendix A, Using the Command Line Interface” in *Sun Java System Application Server Administration Guide*.

- **The `cladmin` command.** The `cladmin` command runs certain `asadmin` commands simultaneously on all application server instances in a cluster. Using the `cladmin` command assures you that all instances in the cluster are being configured identically, so you should use `cladmin`, rather than `asadmin`, whenever possible. Not all `asadmin` commands are supported by `cladmin`.

For more information on `cladmin`, see “Appendix F, Using the `cladmin` Script for Administration” in *Sun Java System Application Server Administration Guide*.

- **The Administration interface.** The Administration interface is a web-based user interface that you can use to configure the Admin Sever and individual application server instances. You can also use it to view the log files of individual application server instances.

For more information on Admin Console, see “Part 1, Server Basics and Administering Global Settings” in *Sun Java System Application Server Administration Guide*.

- **The `clsetup` command.** The `clsetup` command allows you to easily set up a cluster, including initial configuration of the HADB database. For full instructions on the `clsetup` command, see the *Sun Java System Application Server Installation Guide*. Additionally, you should consult the manpages provided with `clsetup`.

NOTE Some aspects of server configuration and corresponding interfaces are still undergoing change. An unstable interface may be removed and replaced with a cleaner and more stable version in a subsequent product release. Most server configuration and administrative interfaces will remain the same or change in a compatible manner; however some aspects may change incompatibly. Product documentation for future releases will clearly describe incompatible changes when they do occur.

Clustering Scenarios

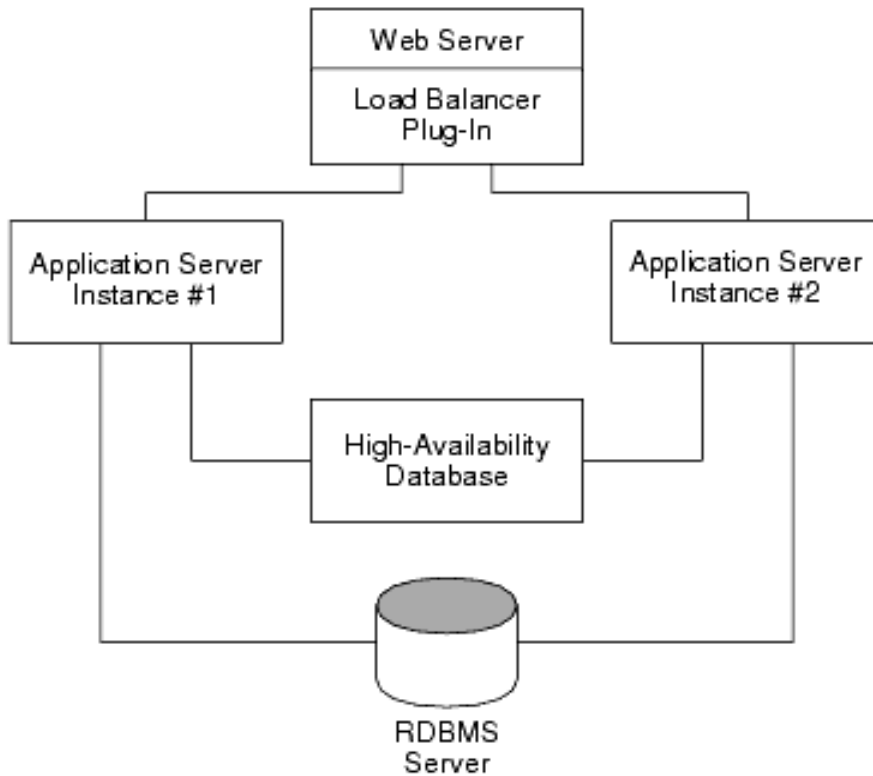
This section describes the following topics:

- [HTTP Clustering Scenario](#)
- [RMI/IIOP Clustering Scenario](#)

HTTP Clustering Scenario

The following diagram shows a simple clustering scenario consisting of a web server with a load balancer plug-in, two Sun Java System Application Server instances configured to use the HADB to store HTTP session data, and a remote database management system (RDBMS) to store application data. Note that your actual operational deployment may vary. For example, you could use a third-party load balancer in place of the load balancer plug-in.

Figure 2-1 Example Clustering Scenario



The following steps describe how an HTTP request is serviced:

1. An incoming client sends an HTTP request to a URL handled by the web server. The web server has been configured to allow the load balancer plug-in to handle the incoming HTTP request.
2. The load balancer plug-in then forwards the request to one of the Sun Java System Application Server instances in the cluster. The plug-in uses sticky round-robin load balancing to determine the target instance.

3. The target instance receives the request forwarded to it from the load balancer plug-in and begins an HTTP session, storing the HTTP session data to the HADB and the J2EE application data to the RDBMS. As the client progresses through the application, HTTP session data is updated and stored to the HADB and application data is updated in the RDBMS.
4. If the instance fails due to a system crash, the load balancer detects that the instance has ceased responding to requests. When subsequent requests come it, the load balancer forwards them to a healthy instance in the cluster.
5. The new target instance retrieves the failed over HTTP session information from the HADB and continues responding to the client's requests, allowing the client to complete the HTTP session with no loss of session data.

For more information about deployment scenarios for the Sun Java System Application Server, see the *Sun Java System Application Server System Deployment Guide*.

RMI/IIOP Clustering Scenario

Sun Java System Application Server provides a highly available J2EE application through the use of load-balancing and failover mechanism on the RMI/IIOP path.

To configure high availability, Application Server instance needs to be configured in an IIOP cluster with the available IIOP listeners that's used for serving RMI-IIOP requests. The IIOP *listeners* that will be part of the IIOP cluster, are called as IIOP *endpoints*.

The following sections describe the changes required to enable load balancing (client-side configuration) and high availability (server-side changes) for J2EE applications on the RMI/IIOP path.

- [Server-side Configuration](#)
- [Client-side Configuration](#)

Server-side Configuration

To enable failover of IIOP requests, you should configure IIOP endpoints to constitute the IIOP cluster in Sun Java System Application Server. IIOP endpoints are defined either using the Administration Console or the command line interface.

Define all the non-SSL endpoints in a cluster to which the RMI/ IIOP requests can be failed over. The `iiop-cluster` property under the `availability-service` element defines the IIOP endpoints. Ensure that `availability-enabled="true"` to enable failover.

The following example shows the IIOP cluster configuration property from the `server.xml` file.

```
<availability-service availability-enabled="true">
  <iiop-cluster>
    <iiop-server-instance name=server1>
      <iiop-endpoint id=s1_ep1 host=trident port=3700 />
      <iiop-endpoint id=s1_ep2 host=trident port=3800 />
      <iiop-endpoint id=s1_ep3 host=trident port=3900 />
    </iiop-server-instance>
    <iiop-server-instance name=server2>
      <iiop-endpoint id=s2_ep1 host=jupiter port=4700 />
      <iiop-endpoint id=s2_ep2 host=jupiter port=4800 />
      <iiop-endpoint id=s2_ep3 host=jupiter port=4900 />
    </iiop-server-instance>
  </iiop-cluster>
</availability-service>
```

Client-side Configuration

To enable load balancing of RMI/IIOP requests, you need to configure the client application. Load balancing is supported for the following two types of RMI/IIOP clients:

- [Stand-alone Client](#)
- [ACC Client](#)

Stand-alone Client

To enable load balancing capabilities in a stand-alone client, the following properties must be defined in the JNDI environment properties or system properties.

- `java.naming.factory.initial`

Set the property to `com.sun.appserv.naming.SLASCtxFactory`.

- `com.sun.appserv.iiop.endpoints`

Specifies the list of IIOP endpoints defined in the `server.xml`.

- `com.sun.appserv.iiop.loadbalancingpolicy`

If the endpoint is specified, then this property is used to specify the load balancing policy. The value used to define this property is based on the JNDI `InitialContext`.

When a RMI/IIOP client invokes an `InitialContext` lookup on a remote object, an Application Server instance (which is part of the IIOP Cluster) is randomly chosen to create the bean. All subsequent operations with this `InitialContext` will be *assigned* or *stuck* to the same Application Server instance.

To implement load balancing capabilities in your RMI/IIOP client, perform the following steps:

1. Set the following JVM properties to configure the ORB.

```
com.sun.CORBA.connection.ORBConnectionFactoryClass=com.sun.appserv.enterprise.iiop.EEIIOPSocketFactory
org.omg.PortableInterceptor.ORBInitializerClass=com.sun.appserv.iiop.EEORBInitializer
```

2. Set the classpath to `appserv-rt.jar` and `appserv-rt-ee.jar`. These jar files are located in the `install_dir/lib` directory.

3. Use the following property of `SlASCtxFactory` class, prior to the instantiation of the `InitialContext`:

```
Properties env = new Properties();

env.put("java.naming.factory.initial",
"com.sun.appserv.naming.SlASCtxFactory");

env.put("com.sun.appserv.iiop.endpoints", "trident:3600,
exodus:3700");

env.put("com.sun.iiop.loadbalancingpolicy", "ic-based");

//create an initial naming context
Context initial = new InitialContext(env);
```

This client code instantiates the JNDI `InitialContext` Object by calling the new `InitialContext(env)`, where `env` is the list of JNDI SPI properties.

ACC Client

To enable load balancing and failover capabilities in an ACC client, the following properties must be defined in `sun-acc.xml`.

- `com.sun.appserv.iiop.endpoints`
- `com.sun.appserv.iiop.loadbalancingpolicy`

Define the load balancing properties in the `sun-acc.xml` file to enable a highly available ACC client. These properties are defined as property elements in the `sun-acc.xml` file.

For example:

```
<client-container>

  <target-server name="saturn" address="saturn" port="3700">

    <property name="com.sun.appserv.iiop.loadbalancingpolicy"
value="ic-based" />

    <property name="com.sun.appserv.iiop.endpoints"
value="qasol-e1:3700,jupiter:3800"/>

  </client-container>
```

Enterprise Features Configuration Tutorial

This chapter includes a tutorial to set up and use features in your Sun Java System Application Server environment. This chapter contains the following sections:

- [Preparing to Use the Tutorials](#)
- [Overview of Tutorial Steps](#)
- [Starting the Server](#)
- [Verifying Server Start-up](#)
- [Creating the loadbalancer.xml File](#)
- [Adding a Cluster to the loadbalancer.xml File](#)
- [Configuring Load Balancing](#)
- [Sample loadbalancer.xml File](#)

Preparing to Use the Tutorials

This guide walks you through setting up a clustered environment, deploying an application to it, and testing the load balancing and HTTP session persistence features with the application. It includes two tutorials: the [Enterprise Features Configuration Tutorial](#), which covers configuring the cluster and load balancer, and the [Cluster JSP Sample Application Tutorial](#), which covers deploying the application and verifying that load balancing and session persistence work.

This section contains the following topics:

- [Installation Requirements](#)
- [Required Steps Before Using the Tutorials](#)

Installation Requirements

The installation configuration suggested here is a basic, single-machine configuration for use with this guide. It is not necessarily the right configuration for your environment. You need to carefully consider your needs before deploying the Sun Java System Application Server in your environment. For more information on recommended configurations, see the *Sun Java System Application Server System Deployment Guide*.

For the purposes of doing the tutorials included in this book, use the following basic configuration:

- One Sun Java System Web Server
- One load balancer plug-in
- One Sun Java System Application Server installation with two server instances configured to use high-availability databases
- One high-availability databases

NOTE	All the instructions in this guide assume that you are using the Sun Java System Web Server. You can also use the Apache web server. For more information on using the Apache web server, see the <i>Sun Java System Application Server Administration Guide</i> .
-------------	--

Required Steps Before Using the Tutorials

Before you start the tutorials, you must have already completed the following steps.

Table 3-1 Steps for Preparing to Use the Tutorials

Step	Location of Instructions
1. Install Sun Java System Web Server	<i>Sun Java System Web Server Installation Guide</i>
2. Install Sun Java System Application Server, including: <ul style="list-style-type: none"> • The load balancer plug-in • The HADB 	<i>Sun Java System Application Server Installation Guide</i>
3. Set up the following for your machine: <ul style="list-style-type: none"> • Shared memory • Communication using rsh or ssh • Environment variables for using the HADB 	<i>Sun Java System Application Server Installation Guide</i>
4. Run <code>clsetup</code> to perform the following configuration: <ul style="list-style-type: none"> • Create and configure two application server instances, <code>server1</code> and <code>server2</code>, in the domain <code>domain1</code> • Create the HADB. • Create the database tables required to store session information in the HADB. • Create a connection pool in the instances • Create a JDBC resource in all the instances. • Configure the session persistence information in the application server instances. • Enable high availability in the application server instances. 	<i>Sun Java System Application Server Installation Guide</i>

Overview of Tutorial Steps

The following table summarizes the steps required in the tutorials to configure your system, deploy an application, and verify the enterprise features. The left column describes the step, and the right column gives the location of the instructions needed to perform the step.

Table 3-2 Steps for Using Tutorials

Step	Location of Instructions
1. Start the Admin Server and application server instances.	“Starting the Server” on page 39
2. Verify that the Admin Server and application server instances are running.	“Verifying Server Start-up” on page 41
3. Create a <code>loadbalancer.xml</code> file.	“Creating the loadbalancer.xml File” on page 45
4. Set up a cluster in <code>loadbalancer.xml</code> .	“Adding a Cluster to the loadbalancer.xml File” on page 46
5. Configure the load balancer in the <code>loadbalancer.xml</code> file.	“Configuring Load Balancing” on page 49
6. Verify your <code>loadbalancer.xml</code> file against the sample <code>loadbalancer.xml</code> file.	“Sample loadbalancer.xml File” on page 53
7. Deploy the sample application to the instances in your cluster using <code>cladmin</code> .	“Deploying the Cluster JSP Sample Application to a Cluster” on page 56
8. Start the application server instances using <code>cladmin</code> .	“Starting the Application Server Instances Using cladmin” on page 60
9. Verify that you deployed your application successfully.	“Verifying Application Deployment” on page 61
10. Add the sample application to the cluster information in <code>loadbalancer.xml</code> .	“Adding the Sample Application to the Cluster” on page 66
11. Apply changes and restart the web server.	“Applying Configuration Changes and Restarting the Web Server” on page 67
12. Run the sample.	“Running the Application” on page 68
13. Verify load balancing.	“Verifying HTTP Load Balancing” on page 73
14. Verify session persistence.	“Verifying HTTP Session Persistence” on page 77
15. Quiesce a server instance.	“Quiescing a Server Instance” on page 78

Starting the Server

Before you can use the tutorials in this guide, you need to start the Admin Server and all the application server instances in your cluster. Your Sun Java System Web Server should also be running.

The most straightforward way to start the Admin Server and the application server instances when all the server instances and the Admin Server are in a single domain is to use the `asadmin` utility's `start-domain` command.

This section contains the following topics:

- [Setting the PATH Variable](#)
- [Running `asadmin start-domain`](#)

Setting the PATH Variable

Before running `asadmin`, for added convenience you can set your PATH variable.

Setting the PATH variable allows you to run `asadmin`, `cladmin`, and the `asant` (the application server's Ant utility) from the command line at any location. Add the following directory to your PATH environment variable:

```
install_dir/bin
```

If you add the Application Server's `bin` directory to your login profile it is automatically added to your environment's PATH setting during login.

For example, add the following to your `.cshrc` file:

```
set path=( /opt/SUNWAppserver7/bin)
```

After you have saved the file, source it:

```
source .cshrc
```

You can test your change by executing the `asadmin` utility at the command prompt. Type:

```
asadmin
```

The following result should appear:

```
Use "exit" to exit and "help" for online help
```

```
asadmin>
```

Type `exit` to quit the `asadmin` interface.

If the command is not found:

- Check that you have updated your PATH setting correctly.
- If you updated your login file (for example, your `.cshrc` file), make sure you sourced the file. You might also need to start a new terminal window.

If you do not set the PATH environment variable, you can still run the commands and utilities from the directory where they are located (*install_dir/bin*), for example:

```
cd /opt/SUNWappserver7/bin
./asadmin
```

Running asadmin start-domain

To start the Admin Server and all the application server instances in it, run the `asadmin start-domain` command. At the command prompt, type:

```
asadmin start-domain --domain domain1
```

`domain1` is the default domain configured when you installed the server and ran the `clsetup` command.

NOTE In the configuration described in this guide, all the instances are in the same domain. If the server instances are not in the same domain, you need to run `asadmin start-domain` for every domain you plan to use. Once you have created a cluster, you can also use `cladmin start-instance` to start all the instances in a cluster. For information on `cladmin` syntax, see [“The cladmin Command Syntax” on page 58](#).

Use the following command to stop both the Admin Server and the application server instances of the initially configured domain:

```
asadmin stop-domain --domain domain1
```

where `domain1` is the name of the administrative domain defined during installation of the application server.

For a full list of `asadmin` commands, see the `asadmin help`. To access help, at the command prompt, type:

```
asadmin
```


Your prompt should change to the `asadmin` prompt, if you have set your `PATH` variable. Type `help` at the prompt to see a list of all `asadmin` commands. To get help on a specific `asadmin` command, at the `asadmin` prompt type the command name followed by the `help` option. For example:

```
asadmin> start-domain --help
```

Verifying Server Start-up

After starting the Admin Server and application server instances, verify that they have started successfully, as described in the following sections:

- [Verifying Admin Server Start-up](#)
- [Verifying Start-up of Server Instances](#)
- [Verifying Instances by Accessing the HTTP Server](#)

Verifying Admin Server Start-up

You can verify that the Admin Server started by accessing the Sun Java System Application Server's Administration interface, or by viewing the Admin Server's event log, as described in the following sections:

- [Accessing the Administration Interface](#)
- [Viewing the Admin Server's Event Log](#)

Accessing the Administration Interface

To verify that the Admin Server started correctly, you can access the Sun Java System Application Server's web-based Administration interface. You can use this interface to administer the Sun Java System Application server instances, though for clustered instances it is more convenient and less error-prone to use the `cladmin` command.

NOTE	For a list of browsers compatible with the Administration interface, see the <i>Sun Java System Application Server Platform Summary</i> .
-------------	---

To access the Administration interface, follow these steps:

1. Open a browser window and specify the Admin Server port.

During installation, the default port number for the Admin Server is set to 4848. If this port was already in use or you selected another port number, then specify that port number.

For example:

`http://test.sun.com:4848`

2. Sign into the Administration interface using the administration user name and password supplied during product installation.

TIP **Forgot the user name or password?** If you do not remember the Admin Server user name that was supplied during installation, try the user name `admin`, the default user name specified in the server configuration dialog during installation.

If you don't remember the administrator's password, look in the `clpassword.conf` file in the *install_config_dir*, created when you installed the server. You must be root to access this file.

TIP **Port not accessible?** If the connection was refused when attempting to contact the Admin Server's Administration interface, it is likely that the Admin Server is not running. Check your start-up procedures and the content of the Admin Server's log file to determine the reason why the server is not running. For instructions on viewing the log file, see [“Viewing the Admin Server's Event Log” on page 42](#).

Once you've authenticated successfully, the initial screen of the Administration interface appears. When you click on an item in the left pane, the corresponding page appears in the right pane.

Viewing the Admin Server's Event Log

You can also look in your Admin Server event log for a server start-up message. To open and view the event log file, follow these steps:

1. Navigate to the server logs for the Admin Server:

`domain_config_dir/domain1/admin-server/logs/`

2. Open `server.log` in an editor.

If your server has been started, you will find successful server startup at the end of the log file:

```
[20/Feb/2003:00:06:00] INFO ( 4318): CORE3274: successful server
startup
```

If you don't see the successful start-up message, it could be because you have opened the event log file prior to the Admin Server completing its startup procedures. Close the log file and open it again to see the very latest event messages.

You can also use the `tail -f` command to view messages in the server log.

```
tail -f server.log
```

The `-f` option leaves the `tail` command running so new log entries are displayed as they are written to the file.

You can also access the Admin Server and the application server instance event log files through the Administration interface. Click the server name in the left pane (Admin Server or the application server instance) and click the Logging tab in the right pane.

For more information, see the *Sun Java System Application Server Administration Guide*.

Verifying Start-up of Server Instances

After running `clsetup`, you should have two server instances in domain1: `server1` and `server2`. This section discusses two ways to verify that those instances exist, and to verify that they are running:

- [Verifying Instances using `asadmin`](#)
- [Verifying Instances by Accessing the HTTP Server](#)

Verifying Instances using `asadmin`

To verify, use the `asadmin list-instances` command:

1. Launch `asadmin` by typing `asadmin` at the command prompt:

```
asadmin
```

Your prompt becomes `asadmin>`

2. Use the `list-instances` command at the `asadmin` prompt to show all current instances.

```
list-instances
```

This command lists the instances and whether or not they are currently running. You should see two running instances, `server1` and `server2`.

Verifying Instances by Accessing the HTTP Server

Another method to determine whether or not an application server instance has started is to access the instance's HTTP server welcome page through a web browser.

Using a browser, access the following location:

```
http://server_instance:server_instance_port_number
```

where *server_instance_port_number* is the HTTP server port number specified during installation or server instance creation.

TIP

If you do not remember the HTTP server port number of your server, you can inspect the application server instance's configuration file to determine the HTTP server port number:

1. Navigate to the `domain_config_dir/domain1/server_instance/config/` directory and open the `server.xml` file in your favorite editor.
2. Look for the `http-listener` element, for example:

```
http-listener id="http-listener-1" address="0.0.0.0"
port="81"...
```

In this case, port 81 is the HTTP port number in use.

If the application server instance is up and running normally, you should see the default HTTP server welcome page in your browser:

TIP **HTTP server welcome page:** The HTTP server welcome page is an HTML page named `index.html` in the application server instance's default document directory. The application server instance's `server.xml` configuration file contains the setting for the default document root of the instance. After installation, the document root for the instance named `server1` is set to `domain_config_dir/domain1/server1/docroot/`. You can find the welcome page at that location.

Creating the loadbalancer.xml File

After you have installed the components, run `clsetup`, and started the Admin Server, you are ready to configure your installation. The first step is to create a cluster.

Since the configuration of the cluster for this guide uses one load balancer plug-in, you must define the cluster in the web server's `loadbalancer.xml` configuration file. This file belongs in the configuration file directory of your web server. The `loadbalancer.xml` file does not exist by default. You must create it.

To create the `loadbalancer.xml` file, follow these steps:

1. Find the sample `loadbalancer.xml` file, called `loadbalancer.xml.example`, located in your web server's `config` directory.

By default, this directory is:

`webserver_install_dir/servers/https-server_name/config`

2. Save a copy of `loadbalancer.xml.example` as `loadbalancer.xml`.

You will edit this copy to include information for your environment, including information to create a cluster and allow load balancing to work.

An example `loadbalancer.xml` file is shown in [Code Example 3-1](#).

Code Example 3-1 The `loadbalancer.xml.example` file

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun ONE Application Server
7.0//EN" "sun-loadbalancer_1_2.dtd">

<loadbalancer>

    <cluster name="cluster1">
```

```
<instance name="instance1" enabled="true" disable-timeout-in-minutes="60"
listeners="<REPLACE_WITH_LISTENER1> <REPLACE_WITH_LISTENER2>" />

<web-module context-root="/abc" enabled="true" disable-timeout-in-minutes="60"
enabled="true" />

<health-checker url="/" interval-in-seconds="10" timeout-in-seconds="30" />

</cluster>

<property name="reload-poll-interval-in-seconds" value="60"/>
<property name="response-timeout-in-seconds" value="30"/>
<property name="https-routing" value="true"/>
<property name="require-monitor-data" value="false"/>

</loadbalancer>
```

Notice the example file refers to the `sun-loadbalancer_1_2.dtd`, also found in the web server's `config` directory.

Adding a Cluster to the loadbalancer.xml File

Next you need to add a cluster, for this tutorial called `cluster1`, to your `loadbalancer.xml` file. The cluster includes two Sun Java System Application Servers or server instances.

Edit your newly-created `loadbalancer.xml` to include the instances and the URLs to the HTTP listeners:

1. Go to your web server's `config` directory and open `loadbalancer.xml` using a text editor.

Notice that the sample file already includes a cluster called `cluster1`.

```
<cluster name="cluster1">
```

2. Add the two application server instances that you created with `clsetup`, `server1` and `server2`, to `cluster1`.

Create an instance subelement of the cluster for each of the instances. The instances need to be enabled (`enabled=true`). You also need to add the URLs of the HTTP listeners. In the case of the default server instance (`server1`) and the second server instance you have added (`server2`), there is one HTTP listener each. You could have more in a different environment.

The sample `loadbalancer.xml` file contains the following lines that you can edit:

```
<instance name="instance1" enabled="true"
disable-timeout-in-minutes="60" listeners="<REPLACE_WITH_LISTENER1>
<REPLACE_WITH_LISTENER2>" />
```

3. Replace “instance1” with “server1” for this tutorial.
4. Replace `<REPLACE_WITH_LISTENER1>` with the listener URL for your server instance, for example:

```
http://test.sun.com:81.
```

The listener URL consists of the server name and port. If the listener has security enabled, the URL starts with HTTPS. If the listener is not secure, the URL starts with HTTP. To find information on the HTTP listener for each Sun Java System Application Server instance:

- k. In the Administration interface, click the instance name to expand the tree view.
 - l. Click HTTP Server to expand it.
 - m. Under HTTP Server, click HTTP Listener.
 - n. The HTTP listener page gives you the port, the server name, and whether security is enabled.
5. Save your changes to `loadbalancer.xml`. Your file now has a line similar to the following:

```
<instance name="server1" enabled="true"
disable-timeout-in-minutes="60" listeners="http://test.sun.com:81"/>
```

6. Change the `disable-timeout-in-minutes` value to 5.

The disable timeout in minutes is the instance-level quiescing period, in which you allow time for sessions to terminate for a server instance. The server is still running, but no requests that would result in new sessions are sent to it, though the server does accept requests that are for existing sessions. In the real world, this number would be quite high. The time specified in the example is 60 minutes. However, because the tutorial requires you to shut down the server, in this tutorial you are asked to set the timeout artificially low.

```
<instance name="server1" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:81"/>
```

7. Add a second instance to the cluster element. You can do that by copying the line above and pasting it below the existing line, then changing the server name and the listener URL. For example:

```
<instance name="server2" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:82"/>
```

8. Save your changes.

So far, your `loadbalancer.xml` file should look similar to the following:

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun ONE Application Server
7.0//EN" "sun-loadbalancer_1_2.dtd">

<loadbalancer>

  <cluster name="cluster1">

    <instance name="server1" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:81"/>

    <instance name="server2" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:82"/>

    <web-module context-root="/abc" enabled="true" disable-timeout-in-minutes="60"
enabled="true" />

    <health-checker url="/" interval-in-seconds="10" timeout-in-seconds="30" />
  </cluster>

  <property name="reload-poll-interval-in-seconds" value="60"/>
  <property name="response-timeout-in-seconds" value="30"/>
  <property name="https-routing" value="true"/>
  <property name="require-monitor-data" value="false"/>
</loadbalancer>
```


You'll add information associated with the deployed application (the sample line that begins with "web-module") during a later procedure.

NOTE Enabling an application server instance in the `loadbalancer.xml` file is different from starting an application server instance. Even if the server is running and you have added it to the cluster, the load balancer does not route requests to the server instance until you enable it in `loadbalancer.xml` by setting the `instance` element's `enabled` attribute to `true`.

Configuring Load Balancing

After creating the `loadbalancer.xml` file and adding a cluster, edit the `loadbalancer.xml` file to include the load balancer settings.

This section contains the following topics:

- [Configuring the Health Checker](#)
- [Enabling Load Balancer Monitoring](#)
- [Other Load Balancer Properties](#)

Configuring the Health Checker

If you enable the health checker, the load balancer periodically checks all the Sun Java System Application Server instances that are flagged as unhealthy to see whether they have become healthy. If an application server instance that was marked as unhealthy has become healthy, the instance is added to the list of healthy instances, and requests are once more routed to it.

You configure the health checker in the `health-checker` element in `loadbalancer.xml`. You configure the listener URL that the health checker pings to see if the server is running, as well as how often the health checker pings the server and how long the health checker waits for a response before marking the server as unhealthy.

The sample `loadbalancer.xml` file includes the following line, which you do not need to edit in order to complete the tasks in this guide.

```
health-checker url="/" interval-in-seconds="10" timeout-in-seconds="30"
/>
```

- The `health-checker url` specifies the URL to ping to determine whether the server instance is healthy (that is, whether it is responding). The example shows a URL of `"/"`. For example, if your listener URL is `http://www.example.com:80`, then a health checker URL of `"/"` pings `http://www.example.com:80/`. For the purposes of this guide, leave the value `"/"`.
- The `interval-in-seconds` attribute specifies the interval between load balancer health checks. You do not need to change the value from the default 10.
- The `timeout-in-seconds` attribute specifies the interval within which the load balancer must receive a response from the pinged server instance in order for the server instance to be considered healthy. You do not need to change the value from the default 30.

For example, using the default values, the health checker follows these steps:

1. The health checker pings the unhealthy listeners using the health checker URL.
2. For each listener, it waits 30 seconds (`timeout-in-seconds`) for a response.

If the server responds within 30 seconds, the listener is marked healthy. If the server does not respond, the health checker considers the server to still be unhealthy.

3. The health checker waits ten seconds (the `interval-in-seconds`) before starting another cycle of health checks.

If there are no unhealthy instances when the health checker starts, it skips the first two steps.

Enabling Load Balancer Monitoring

The load balancer plug-in uses the web server logging mechanism to write log messages. When monitoring is enabled, the following information is logged to the web server log file:

- Start/stop information for every request
- Failover request information when a request is failed over from an unhealthy to a healthy instance
- List of unhealthy instances logged at the end of every health checker cycle

CAUTION When logging is enabled on the load balancer plug-in, and if the web server logging level is set to `DEBUG` or to print verbose messages, the load balancer writes HTTP session IDs in the web server log files. Therefore, if the web server hosting the load balancer plug-in is in the DMZ, we recommend that you do not use the `DEBUG` or similar log level in production environments.

If you must use the `DEBUG` logging level, then you should turn off load balancer logging by setting `require-monitor-data` property to `false` in `loadbalancer.xml`.

To monitor load balancing, you need to:

- Enable monitoring in `loadbalancer.xml`
- Configure the web server to use verbose logging.

To enable monitoring in `loadbalancer.xml`, follow these steps:

1. Open `loadbalancer.xml` using a text editor.
2. Find the following line (it was part of the example `loadbalancer.xml` file):

```
<property name="require-monitor-data" value="false"/>
```
3. Change “false” to “true” to enable monitoring.
4. Save your changes and exit the file.

In addition, to change the default web server log level, follow these steps:

Turn on the Sun Java System Web Server’s verbose logging:

1. Go to the web server’s `config` directory. If you installed your Sun Java System Web Server in the default location, the path will be:

```
/web_server_install_dir/https-server_name/config
```

2. Open the `magnus.conf` file for editing.
3. Add the following line:

```
LogVerbose on
```
4. Save the changes to the file.
5. Restart the web server to apply the changes.

Other Load Balancer Properties

The example `loadbalancer.xml` contains the following additional load balancer properties:

```
<property name="reload-poll-interval-in-seconds" value="60"/>
<property name="response-timeout-in-seconds" value="30"/>
<property name="https-routing" value="true"/>
```

For this tutorial, you do not need to change the default values. For more information on these properties, see the following sections:

- [Dynamic Reconfiguration using the Reload Poll Interval](#)
- [Response Timeout](#)
- [HTTPS Routing](#)

Dynamic Reconfiguration using the Reload Poll Interval

After the initial configuration, the load balancer plug-in detects changes to its configuration and loads them automatically. It detects changes by looking at the time stamp on the `loadbalancer.xml` file. If the time stamp has changed, the load balancer automatically reconfigures itself. The reload poll interval specifies how often the load balancer checks the time stamp.

NOTE

- If the changes to the `loadbalancer.xml` file are not in the correct format as specified in the `sun-loadbalancer_1_0.dtd` file, the reconfigure operation fails. The failure notice is logged to the web server's error log files.

The load balancer will continue to use the old configuration that's already loaded in memory.

- If the load balancer encounters a hard disk read error while attempting to reconfigure itself, it uses the configuration that is currently in memory.

If a disk read error is encountered, a warning message is logged to the web server's error log file.

The Sun Java System Web Server's error log file can be found at:
`web_server_install_dir/web_server_instance/logs/`.

Response Timeout

The response timeout designates how long the server waits for the instance to respond to a request. After the specified number of seconds, if the instance has not responded an error message is sent to the browser.

HTTPS Routing

When HTTPS routing is disabled, HTTPS listeners listed in the `loadbalancer.xml` file are ignored, and only HTTP listeners are used for load balancing. When HTTPS routing is enabled, HTTPS requests are failed over only to HTTPS ports.

Sample loadbalancer.xml File

The following sample `loadbalancer.xml` file includes one cluster of two instances. Check the syntax of your `loadbalancer.xml` file against this example.

Code Example 3-2 The loadbalancer.xml File

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun ONE Application Server
7.0//EN" "sun-loadbalancer_1_2.dtd">

<loadbalancer>
  <cluster name="cluster1">
    <instance name="server1" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:81"/>
    <instance name="server2" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:82"/>
    <web-module context-root="/abc" enabled="true" disable-timeout-in-minutes="60"
enabled="true" />
    <health-checker url="/" interval-in-seconds="10" timeout-in-seconds="30" />
  </cluster>
  <property name="reload-poll-interval-in-seconds" value="60"/>
  <property name="response-timeout-in-seconds" value="30"/>
  <property name="https-routing" value="true"/>
  <property name="require-monitor-data" value="true"/>
```

</loadbalancer>

Now that you have configured the enterprise features, you can deploy and run the sample application. Continue on to [Chapter 4, “Cluster JSP Sample Application Tutorial.”](#)

Cluster JSP Sample Application Tutorial

This chapter demonstrates simple HTTP failover using the load balancer plug-in. It does not describe all the features supported by the HTTP load balancer plug-in. For information on the other features offered by the load balancer plug-in, see *Sun Java System Application Server Administration Guide*.

This chapter contains the following sections:

- [Preparing to use the Cluster JSP Sample Application Tutorial](#)
- [Deploying the Cluster JSP Sample Application to a Cluster](#)
- [Starting the Application Server Instances Using cladmin](#)
- [Verifying Application Deployment](#)
- [Adding the Sample Application to the Cluster](#)
- [Applying Configuration Changes and Restarting the Web Server](#)
- [Running the Application](#)
- [Verifying HTTP Load Balancing](#)
- [Verifying HTTP Session Persistence](#)
- [Quiescing a Server Instance](#)

Preparing to use the Cluster JSP Sample Application Tutorial

Before you use this tutorial, you must have already:

- Successfully completed all the tutorial steps in [Chapter 3, “Enterprise Features Configuration Tutorial.”](#)
- Made a copy of the sample applications, if necessary.

You may need to make a copy of the samples if you are either sharing your application server installation with other users or your system user ID does not have write permissions to the area in which the application server is installed.

To copy the samples, copy the *install_dir/samples* directory to a location in which your user ID has write permissions.

For example:

```
cp -r install_dir/samples user_sample_directory
```

The *samples* directory contains the subdirectory *ee-samples* which contains the sample used in this guide.

If you are using a copy of the samples, as you proceed with this guide, replace *install_dir/samples/* with the location of your own copy of the sample applications.

Deploying the Cluster JSP Sample Application to a Cluster

The Cluster JSP sample application demonstrates how a request to a JSP can be load balanced between application servers in a cluster. It has a shopping cart which demonstrates how HTTP session information persists, even if an application server has an outage.

The sample application comes with a WAR file that is ready to deploy as a web application. You do not need to compile or assemble the application.

NOTE All applications that use session failover must be distributable. The `clusterjsp` sample application is already distributable, as you can see if you look in its `web.xml` file.

To see this:

1. Go to the Cluster JSP sample's `src` directory:

```
cd /install_dir/samples/ee-samples/clusterjsp/src
```

2. Look at the `web.xml` file. You see the following code:

```
<web-app>

    <display-name>clusterjsp</display-name>

    <distributable/>
```

The `web-app` element has the `distributable` subelement specified. No changes are required for the application code.

To deploy the sample application to a cluster, you must first deploy it to every instance in the cluster.

Use the `cladmin` command to deploy an application to all instances in your cluster at once. The `cladmin` command runs an `asadmin` command on all instances in a cluster simultaneously. The `cladmin` command is located in `install_dir/bin`.

This section contains the following topics:

- [Input Files for the cladmin Command](#)
- [The cladmin Command Syntax](#)
- [Running cladmin deploy](#)
- [The asadmin Commands Supported by the cladmin Command](#)
- [Requirements and Limitations](#)

Input Files for the cladmin Command

The `cladmin` command uses two input files: `clinstance.conf` and `clpassword.conf`.

- `clinstance.conf`: This file contains information about the application server instances that are part of the cluster.

- `clpassword.conf`: This file contains the administration server password and is pre-populated with the correct password during a standard installation.

These files are located in the *install_config_dir*. Since you used these files to run `clsetup` earlier, these files should have the correct values for your environment.

Because many of the values you would have to specify to run `cladmin deploy` are included in these files, you can run the `deploy` command with a minimum of options.

The cladmin Command Syntax

The syntax of the `cladmin` command is as follows:

```
cladmin [--help] [--instancefile instance_file_location] [--passwordfile
password_file_location] asadmin_command
```

where:

- *instance_file_location* is the location of the input file `clinstance.conf`
- *password_file_location* is the location of the input file `clpassword.conf`
- *asadmin_command* is the `asadmin` command that you want to run on the application server instances in the cluster.

If the input files are located in the default location `/etc/opt/SUNWappserver7`, you can omit the `instancefile` and `passwordfile` options and run the command as follows:

```
cladmin asadmin_command
```

Running cladmin deploy

To deploy the application to all the instances in cluster, type:

```
cladmin deploy filepath
```

You can deploy the sample as a web application using the WAR file provided with the sample, the *filepath* is the path to the WAR file.

For example, if your Cluster JSP application is in the default location, type:

```
cladmin deploy /opt/SUNWappserver7/samples/ee-samples/clusterjsp/clusterjsp.war
```

If you have set your PATH variable, you can also go to the directory where the WAR file is located and deploy the application from there. For example:

```
cd install_dir/samples/ee-samples/clusterjsp
cladmin deploy clusterjsp.war
```

When you run the `cladmin deploy` command, you see messages as the application is deployed to each instance in the cluster.

If you encounter errors, consult the log file at `/var/tmp/cladmin.log` for more information.

NOTE If you are running as root and have not set the PATH variable for root, you may not be able to run `cladmin` from any directory. Change to the *install_dir/bin* directory, then type `./cladmin asadmin_command` at the command prompt.

The asadmin Commands Supported by the cladmin Command

You can use the `cladmin` command to run the following `asadmin` commands simultaneously for each application server instance in a cluster, except for `configure-session-persistence`:

Table 4-1 The asadmin Commands Supported by cladmin

Command	Use
<code>start-instance</code>	Starts the application server instances
<code>stop-instance</code>	Stops the application server instances
<code>deploy</code>	Deploys an EJB, WEB, connector, appclient, or application component that is in the directory to the application server instances
<code>undeploy</code>	Removes the deployed component from the application server instances
<code>create-jdbc-resource</code>	Creates a JDBC resource for the application server instances
<code>create-jdbc-connection-pool</code>	Creates a JDBC connection pool for the application server instances

Table 4-1 The asadmin Commands Supported by cladmin

Command	Use
configure-session-persistence	Configures session persistence for the application server instances. This command needs to be run only once for all instances of an Application Server running on the same machine.
delete-jdbc-resource	Deletes the JDBC resource for the application server instances
delete-jdbc-connection-pool	Deletes the JDBC connection pool for the application server instances

For more information about the `cladmin` command, see the *Sun Java System Application Server Administration Guide*.

Requirements and Limitations

The `cladmin` command has the following requirements and limitations:

- All instances that are part of the cluster must have the same password for the administrator. The user name can be specified in the property name, `user`, in `clinstance.conf` file.
- Before running the command you must start the Admin Servers associated with all application server instances that are part of the cluster.
- The values specified in the input files will be same for all the instances in a cluster. The `cladmin` command is not designed to set up each instance with different values. For example, this command cannot create a JDBC Connection Pool with different settings for each instance.

Starting the Application Server Instances Using cladmin

After you deploy the sample application, verify that the server instances are running.

1. At the command prompt, type:

```
asadmin list-instances
```

The command displays the instances and whether they are running or not.

2. If the instances are not running, start them using `cladmin` as follows:

```
cladmin start-instance
```

You see messages as each server instance is started.

Verifying Application Deployment

Once the application is deployed, you can run it in the Sun Java System Application Server instance to test the deployment. To run the sample application, follow these steps:

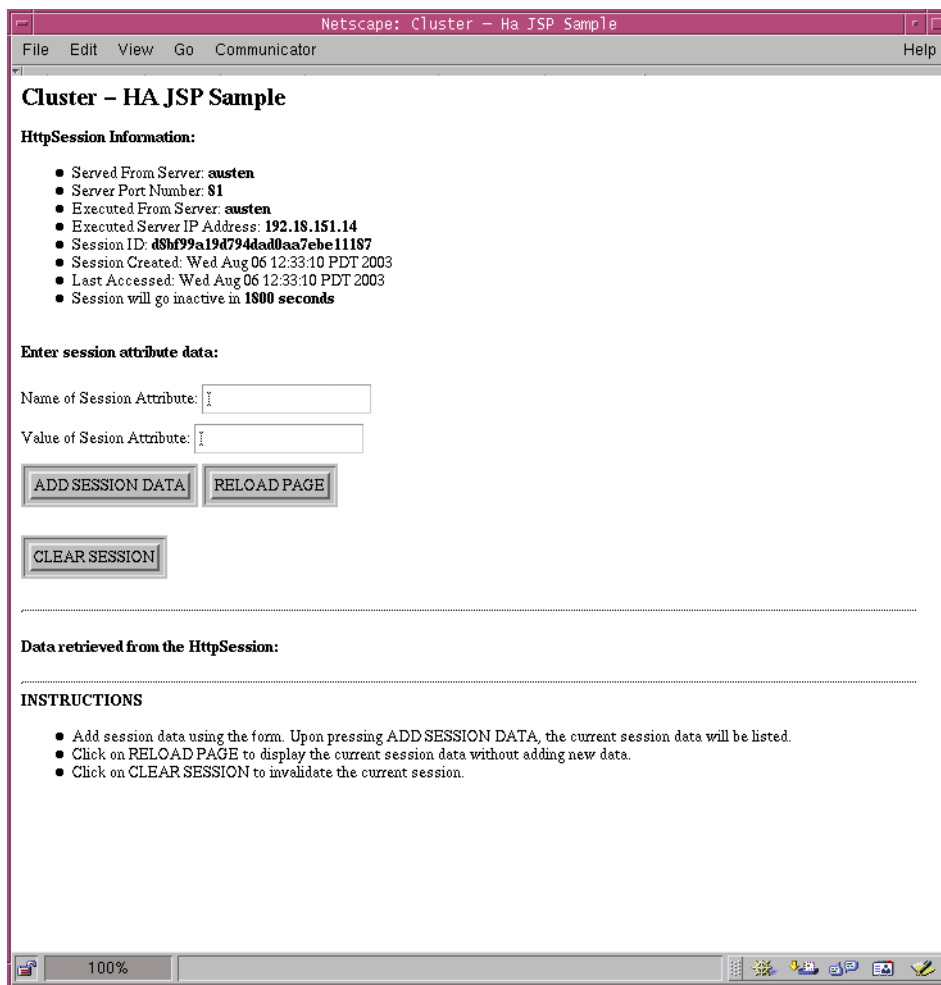
1. From a browser, access the following URL:

```
http://host:application_server_instance_port/clusterjsp
```

For example:

```
http://test.sun.com:81/clusterjsp
```

2. You see the page for this sample application.

Figure 4-1 Cluster JSP Sample Application Page

If you encounter an error, see [“Troubleshooting Deployment to an Application Server Instance”](#) on page 65.

3. Enter some information in the session attribute fields and click ADD SESSION DATA.

The application displays your data.

4. To verify deployment to the second instance, type the other instance's port and hostname in the URL:

`http://host:application_server_instance_port/clusterjsp`

For example:

`http://test.sun.com:82/clusterjsp`

These URLs verify that the application was deployed to the application server instances. The application has not yet been added to the cluster in `loadbalancer.xml`, so you cannot yet verify it through the web server.

Monitoring the Sample in the Application Server

You can monitor the sample application by looking in the event log file of the application server instance. Each application server instance has its own event log.

This section contains the following topics:

- [Using the Administration Interface to View Logs](#)
- [Viewing Logs Using the tail Command](#)
- [Application-Generated Messages in the Event Log](#)
- [Application-Generated Messages in the Access Log](#)

Using the Administration Interface to View Logs

To use the Administration interface to view server instance log files:

1. Access the Administration interface.
2. In the left pane, click the instance for which you want to check logs.
3. Click the Logging tab.
4. Click View Event Log
5. To refresh the log view, click OK.

If you would like to see more than 25 log entries, simply enter a larger number in the "Number of errors to view?" area and click OK to refresh the log display.

To use the Administration interface to view the HTTP access log for an application server instance:

1. Access the Administration interface.

2. In the left pane, click the instance for which you want to check logs.
3. Click the Logging tab.
4. Click View HTTP Access Log.

The HTTP access log for a server instance shows the accesses to the sample application made for the server instance.

The name of the access log is `access`. By default, the access log file is located in the same directory as the server event log:

```
domain_config_dir/domain1/server1/logs/
```

Viewing Logs Using the tail Command

Using the `tail -f` command to monitor log files lets you see messages automatically as they are logged. In the Administration interface you must click a button to reflect new information on the page.

To use the `tail -f` command, follow these steps:

1. Navigate to the log directory of the server instance you want to monitor, for example:

```
cd domain_config/domain1/server1/logs/
```

2. Run `tail` on the server event log file:

```
tail -f server.log
```

The `-f` option leaves the `tail` command running so new log entries are displayed as they are written to the file.

Application-Generated Messages in the Event Log

When your application writes information to `stdout` and/or `stderr`, by default, this information is recorded in the server instance's event log as INFO-level messages with the message ID `CORE3282` (`stdout`) and `CORE3283` (`stderr`). While running the Cluster JSP sample, a number of application generated messages are written the server event log. The following excerpt provides a snapshot of some of these messages:

For example:

```
[13/Aug/2003:17:32:28] INFO ( 9657): CORE3282: stdout: No parameter entered  
for this request
```

```
[13/Aug/2003:17:32:34] INFO ( 9657): CORE3282: stdout: Add to session:  
name1 = 1
```



```
[13/Aug/2003:17:32:45] INFO ( 9657): CORE3282: stdout: Add to session:
name2 = 2
```

```
[13/Aug/2003:17:32:52] INFO ( 9657): CORE3282: stdout: Add to session:
name3 = 3
```

These messages show the changes in values as data is entered into the application's user interface.

Application-Generated Messages in the Access Log

When you run the Cluster JSP sample, access log messages are logged to the application server instance that served the request:

```
192.18.151.14 - - [12/Aug/2003:15:34:52 -0700] "GET /clusterjsp/ HTTP/1.0"
302 1086
```

```
192.18.151.14 - - [12/Aug/2003:15:34:52 -0700] "GET /clusterjsp/HaJsp.jsp
HTTP/1.0" 200 1578
```

Troubleshooting Deployment to an Application Server Instance

The most common problems when attempting to run this sample application are described in the following table. The left column shows the symptom, the middle column the probable cause of the problem, and the right column shows suggested solutions to try.

Table 4-2 Troubleshooting Deployment to Application Server Instance

Symptom	Probable cause	Solution
Connection failure when attempting to access first page.	Application server not started. Wrong port specified in URL.	Ensure that the application server has been started. Determine the correct HTTP server port number. See "Starting the Server" on page 39 for more details.
404 error when accessing main page.	Application not deployed.	See "Deploying the Cluster JSP Sample Application to a Cluster" on page 56 .

When troubleshooting problems, it is very important that you monitor the application server log file. You may also find it useful to review the HTTP access log file to verify that HTTP requests are arriving at the application server as expected.

Adding the Sample Application to the Cluster

Once you have deployed the sample application to the two server instances in your cluster, you must add the application to the cluster you created in

`loadbalancer.xml`:

1. Go to your web server's `config` directory and open `loadbalancer.xml` using a text editor.
2. The sample `loadbalancer.xml` file contains the following line:

```
<web-module context-root="/abc" enabled="true"
disable-timeout-in-minutes="60" enabled="true" />
```

Change this line to reflect the `clusterjsp` sample application:

```
<web-module context-root="clusterjsp" enabled="true"
disable-timeout-in-minutes="60"/>
```

This line adds the `clusterjsp` application to the cluster and enables it.

The context root is the same value as is set for the application in the Sun Java System Application Server instance's `server.xml` file.

The `disable-timeout-in-minutes` attribute is set to 60. If the application is disabled, there is a 60-minute application quiescing period for the server to finish servicing any outstanding requests for the application. Earlier we set the instance-level `disable-timeout-in-minutes`, which controls the quiescing period for a server instance.

3. Save your changes.

Your `loadbalancer.xml` file should now be similar to the following file:

Code Example 4-1 The `loadbalancer.xml` File with Sample Application Configured

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun Java
System Application Server 7.0//EN" "sun-loadbalancer_1_0.dtd">

<loadbalancer>

    <cluster name="cluster1">
```

```

        <instance name="server1" enabled="true"
disable-timeout-in-minutes="5" listeners="http://test.sun.com:81"/>
        <instance name="server2" enabled="true"
disable-timeout-in-minutes="5" listeners="http://test.sun.com:82"/>
        <web-module context-root="clusterjsp" enabled="true"
disable-timeout-in-minutes="60"/>
        <health-checker url="/" interval-in-seconds="10"
timeout-in-seconds="30" />
    </cluster>
    <property name="reload-poll-interval-in-seconds" value="60"/>
    <property name="response-timeout-in-seconds" value="30"/>
    <property name="https-routing" value="true"/>
    <property name="require-monitor-data" value="true"/>
</loadbalancer>

```

Applying Configuration Changes and Restarting the Web Server

After you have completed your `loadbalancer.xml` edits, you need to start your web server, or if your web server is already running, you need to apply your configuration changes and restart the web server.

To start your web server:

1. Go to `web_server_install_dir/https-instance_name`.

For example:

```
/usr/iplanet/servers/https-test.sun.com
```

2. Type `./start` to start the server

You may also need to start the web server's administration server, located at `web_server_install_dir/https-admin-serv`.

To apply changes to a running web server and restart it:

1. Access the Server Manager for the web server by typing the appropriate URL into your web browser.

`http://web_server:web_server_admin_port`

For example:

`http://test.sun.com:8888`

2. Choose your server from the server manager and click Manage.
The Server Manager for the web server instance is displayed.
3. A message warns that you have made manual changes to your configuration files and you have to apply changes.
4. Click the Apply link in the upper right of the page.
5. Click Apply Changes to apply the changes and restart the server.

For subsequent `loadbalancer.xml` changes, you do not need to apply changes and restart the server. All subsequent changes are loaded automatically because the reload poll interval is set. For more information, see [“Dynamic Reconfiguration using the Reload Poll Interval” on page 52](#).

Running the Application

After you have deployed the application to all instances and updated the `loadbalancer.xml` file, test the application on the web server to make sure it works.

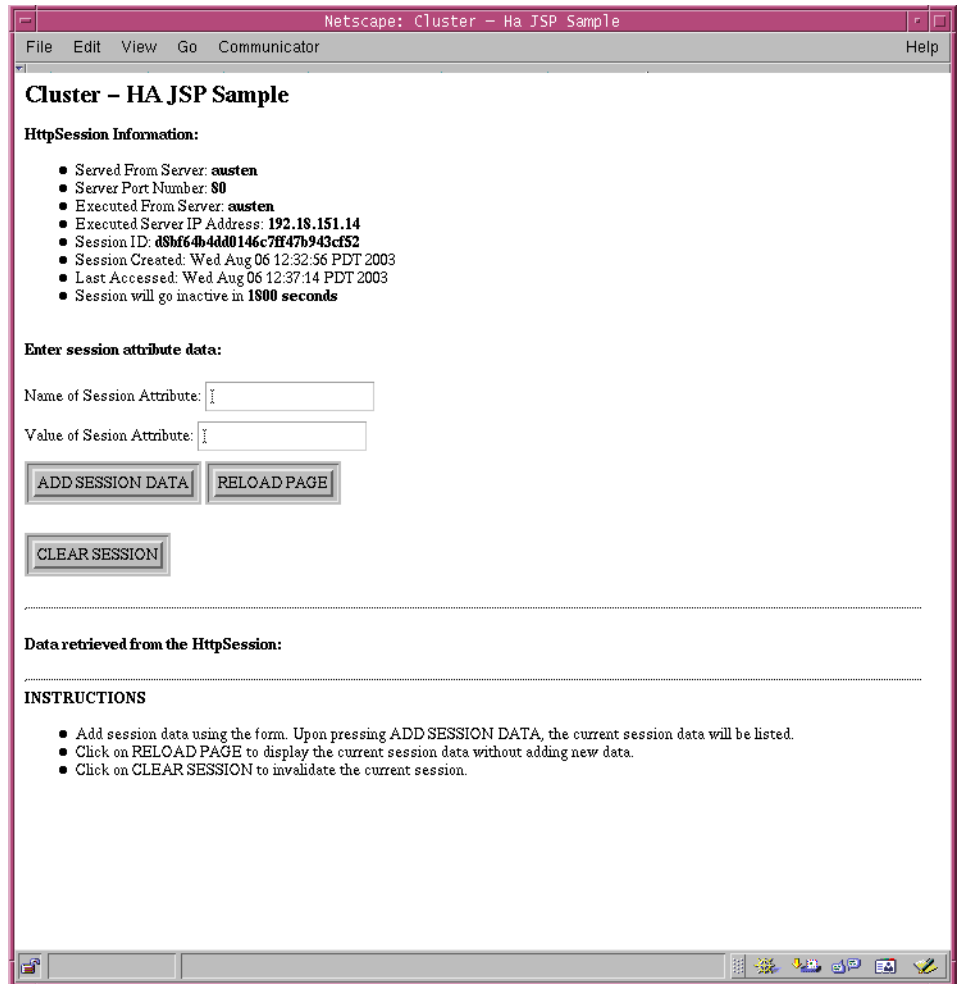
1. From a browser, access the following URL:

`http://host:web_server_port/clusterjsp`

For example:

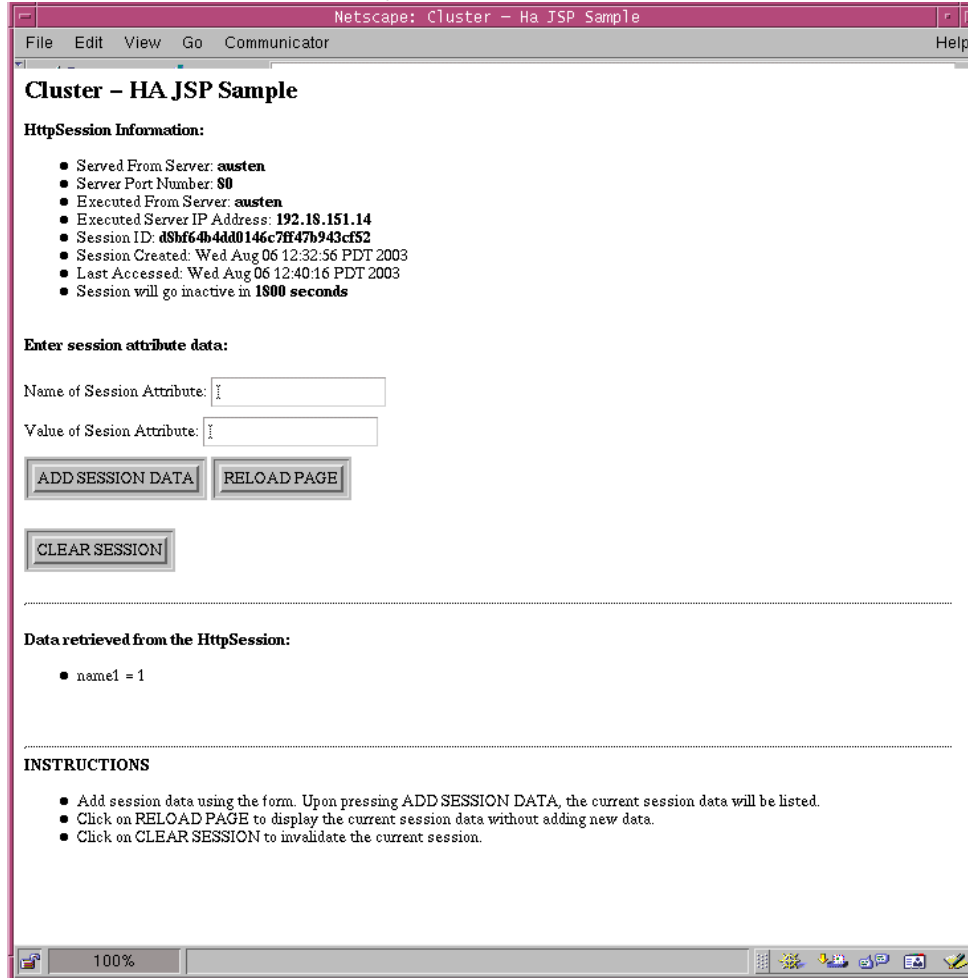
`http://test.sun.com:80/clusterjsp`

You see the page for this sample application.

Figure 4-2 Cluster JSP Sample Application Page

Your server information is listed at the top of the page, along with a unique Session ID. If you are accessing this application on the application server instance directly, the port information at the top of the page is the application server information. If you are accessing a clustered application through the web server, the port information is the web server information.

2. Enter a name and value for the session attribute and click Add Session Data. Your session data is displayed.

Figure 4-3 Session Data Displayed in Cluster JSP

Notice the Session ID at the top of the page. If you entered the session attributes and clicked ADD SESSION DATA before the session timed out, the Session ID at the top of the page will be the same as the first access.

In the example above, the Session ID d8bf64b4dd0146c7ff47b943cf52 is the same in both cases.

If the session timed out, the Session ID will be different.

3. Enter a new session attribute name and value. If the session has not timed out, all attributes are listed in the Data Retrieved from HttpSession area.

For example, if you enter session attribute data twice more before the session timed out, a page similar to the following is displayed:

Figure 4-4 Cluster JSP Sample Application with Multiple HTTP Session Attributes

Cluster - HA JSP Sample

HttpSession Information:

- Served From Server: **austen**
- Server Port Number: **\$0**
- Executed From Server: **austen**
- Executed Server IP Address: **192.18.151.14**
- Session ID: **d5bf64b4dd0146c7ff47b943cf52**
- Session Created: Wed Aug 06 12:32:56 PDT 2003
- Last Accessed: Wed Aug 06 12:42:49 PDT 2003
- Session will go inactive in **1800 seconds**

Enter session attribute data:

Name of Session Attribute:

Value of Session Attribute:

Data retrieved from the HttpSession:

- name2 = 2
- name3 = 3
- name1 = 1

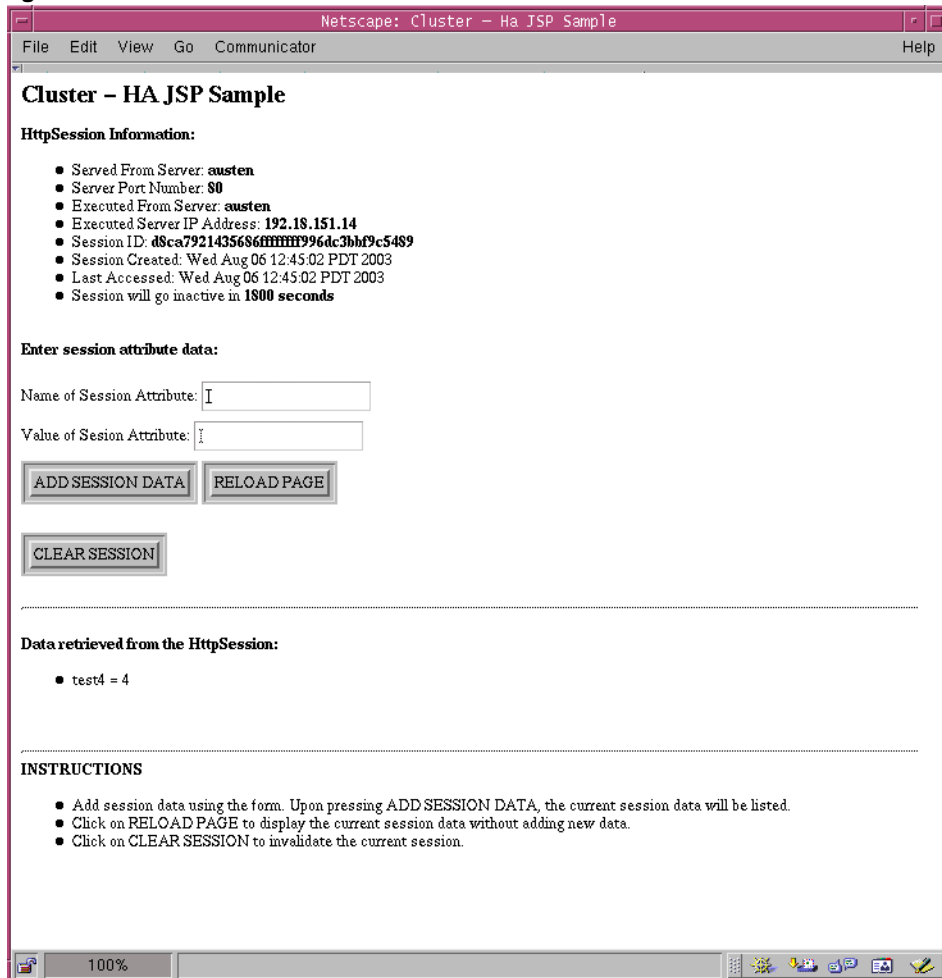
INSTRUCTIONS

- Add session data using the form. Upon pressing ADD SESSION DATA, the current session data will be listed.
- Click on RELOAD PAGE to display the current session data without adding new data.
- Click on CLEAR SESSION to invalidate the current session.

4. If you wait until the session times out (1800 seconds) and enter new session attribute data, you see only the latest data you entered, and a new session ID at the top of the page.

You can also simulate a timeout by clicking **CLEAR SESSION**, which will clear your session data and give you a new session ID.

Figure 4-5 Cluster JSP with New HTTP Session Information



Notice a new Session ID (in this example, d8ca7921435686ffffff996dc3bbf9c5489) appears at the top of the page and that only the latest information entered (in this example, test4=4) appears in the Data Retrieved from HttpSession section.

Verifying HTTP Load Balancing

Now that you have run the sample application and are familiar with how the session information is presented, you can demonstrate that your load balancer is working. If not, you can troubleshoot your load balancer.

This section contains the following topics:

- [Steps for Verifying Load Balancing](#)
- [Troubleshooting the Load-Balancer Plug-in](#)

Steps for Verifying Load Balancing

You can demonstrate that the load balancer is working by following these steps.

1. Open two separate browsers and run the application from your web server by typing `http://web_server_name:web_server_port/clusterjsp`. For example:

```
http://test.sun.com:80/clusterjsp
```

Because session data is sticky when you use the load balancer, if you open a browser window and access the application, all reloads or other operations from that browser are considered to be in the same session and will be serviced by a single Sun Java System Application Server instance.

In order to verify that load balancing is working for the application, open a second browser and access the application. Because the session data is stored in cookies, you must either open the second browser on a different machine, or use different browser software on the same machine. Check the Session ID to make sure both browser windows have separate session data.

2. Once you have accessed the application in two separate session windows, open another browser window and look at the web server error log.
 - a. Access your Sun Java System Web Server's Administration Server with the URL described in your web server documentation, for example:

```
http://test.sun.com:8888
```

- b. Choose your web server instance and click Manage.

- c. In the Server Manager for your web server instance, click the Logs tab.
- d. In the left pane, click View Error Logs.
- 3. Look for entries with RequestExit in them.

You can either scan for them visually or enter RequestExit into the “Only show entries with” field.
- 4. The entries with RequestExit should show HTTP listener IDs for both Sun Java System Application Server instances, so you can see that both application server instances are serving requests.
- 5. To verify that the Sun Java System Application Server instance serviced the request, check the HTTP access logs for each application server instance.

You should see the requests for the `clusterjsp` application. For more information on checking the access logs, see [“Using the Administration Interface to View Logs” on page 63](#).

NOTE In order to see the RequestExit messages, you must have enabled verbose logging for your web server. For instructions, see [“Enabling Load Balancer Monitoring” on page 50](#).

If you are having trouble getting your load balancer to work, see the following section on troubleshooting your load-balancer plug-in.

Troubleshooting the Load-Balancer Plug-in

Use the following table to help you troubleshoot load balancing. The symptom is described in the left column, the probable cause in the middle column, and the solution in the right column.

Table 4-3 Troubleshooting Load Balancing

Symptom	Probable cause	Solution
Connection failure when attempting to access application	Web server is not started Wrong port specified in URL.	Ensure that the web server has been started. Determine the correct web server port number. See your web server documentation for more details.

Table 4-3 Troubleshooting Load Balancing

Symptom	Probable cause	Solution
404 error when accessing application	Application is not deployed. Error in your <code>loadbalancer.xml</code> file	See “Deploying the Cluster JSP Sample Application to a Cluster” on page 56 . Errors in your <code>loadbalancer.xml</code> file are noted in the web server’s error log. For more information, see “Finding Errors in Your loadbalancer.xml File” on page 75 .
Error page when accessing application	One or more of the servers in the cluster is not responding.	Make sure that the server instances in your cluster are running. Check your error log to see if the health checker has found an unhealthy instance. For more information, see “Using the Health Checker” on page 76 .

Finding Errors in Your `loadbalancer.xml` File

If you cannot reach the sample application through the web server URL, you might have a problem with your load balancer. If you can reach the applications through the individual application servers, you can determine that the problem is not with the application server instances.

Because the load balancer plug-in logs messages to the web server log files, you should check the web server error log files for more information.

You can view the error logs for the Sun Java System Web Server through the Server Manager:

1. Access your Sun Java System Web Server’s Administration Server with the URL described in your web server documentation, for example:
`http://test.sun.com:8888`
2. Choose your web server instance and click Manage.
3. In the Server Manager for your web server instance, click the Logs tab.
4. In the left pane, click View Error Logs.
5. The default view shows 25 messages; you can increase the number if necessary.

If you are experiencing problems with the load balancer, you see messages that include “Failed to initialize load balancing subsystem.”

If the problem is that the your `loadbalancer.xml` file has errors, you may see a parser error that points you to the line in your `loadbalancer.xml` file that contains the error. For example, you might see something like:

```
[17/Jun/2003:09:57:44] catastrophe ( 5643): LBConfigParser.cpp@434:
reports: lb.configurator: CNFG1000: Parsing of file
:/usr/iplanet/servers/https-test.sun.com/config/loadbalancer.xml

Failed at line : 7 and column : 3.The error message is : No character data
is allowed by content model
```

This message points you to the line with the error in the `loadbalancer.xml` file.

Correct the problem in the `loadbalancer.xml` file, apply changes and restart the web server, and try again.

Using the Health Checker

If your load balancer is running correctly, and you have set your web server's `LogVerbose` to on, you see the health checker working in the web server's log files:

```
[09/Apr/2003:13:36:02] verbose ( 7576): HealthChecker.cpp@153: reports:
lb.monitor: HLCK1006:UnhealthyInstances cluster1 1049920562631
NoUnhealthyInstances
```

For instructions about turning on verbose logging, see [“Enabling Load Balancer Monitoring” on page 50](#).

Regardless of whether you are using verbose logging, if one of your instances becomes unhealthy, the load balancer will flag it as unhealthy when an access attempt fails. If it remains unhealthy, the health-checker flags it as unhealthy again.

For example:

```
[17/Jun/2003:10:37:27] warning ( 5700): reports: lb.runtime: RNTM2024:
Daemon http://test.sun.com:81 is unhealthy.

[17/Jun/2003:10:37:27] warning ( 5700): reports: lb.healthchecker:
HLCK3003: Listener: http://test.sun.com:81 is detected to be still
unHealthy in cluster: cluster1
```

The health checker continues to monitor the unhealthy instance and flags it as healthy again once it is running.

Verifying HTTP Session Persistence

Once you have verified that load balancing works, you can verify HTTP session persistence. In session persistence, if one Sun Java System Application Server instance fails for some reason, the second instance will pick up the session and process the request.

NOTE Session persistence does not work between clusters, only between instances in a cluster.

To demonstrate that HTTP session persistence is working:

1. Open a browser window.
2. Access the application from the web server. For example:
`http://test.sun.com:80/clusterjsp`
3. View the web server error log to see which server serviced the request (again, look for `RequestExit` entries).
4. Shut down the Sun Java System Application Server instance that is serving the page.

You can use the Administration interface to shut it down:

- a. Access the Administration interface using the administration port number, for example:
`http://test.sun.com:4848`
 - b. In the left pane, click the server instance you want to stop.
 - c. Click Stop in the right pane.
5. Reload the Cluster JSP sample application page.

The session ID and session attribute data is retained.

6. Check the web server error log. Notice that the other Application Server instance is now servicing the request.

Session information should be saved after a server goes offline. The only time you might lose session data during a failure is if the failure occurs while the application is transferring data. In that case, you may see slightly older session data.

Quiescing a Server Instance

In the real world, you would not shut down a server without attempting to have it complete existing sessions. Instead, you would shut down the server in a phased manner, in a process called quiescing.

To quiesce an application server instance, follow these steps:

1. Open a browser window and access the Cluster JSP application through the web server port. Keep this window open.
2. Determine which application server instance is serving the request.

Open a second browser window and access the server1 application server instance HTTP access log in the Administration interface. Look in the log for an access of the Cluster JSP application timed when you sent the request. If you see the access in the server1 log, then the server1 application server instance is serving the request. If you do not see such an entry in the access log, check the access log for the server2 application server instance. Leave the access log window open.

3. Once you have verified the application server instance that is serving the request, disable that server instance in the `loadbalancer.xml` file.

Change `enabled="true"` to `enabled="false"` for the correct server instance. For example:

```
<instance name="server1" enabled="false"
disable-timeout-in-minutes="5" listeners="http://test.sun.com:81"/>
```

4. Save your `loadbalancer.xml` changes.
5. Wait for the reload poll interval to pass, so that your new configuration is loaded.

For this tutorial, the reload poll interval is set to 60 seconds, so you only need to wait a minute.

6. Once the reload poll interval has passed and your new configuration is loaded, the load balancer plug-in no longer sends requests that do not have existing sessions to that server instance. However, requests with existing sessions are still forwarded to the server instance for the quiescing period.

To see this, in your open Cluster JSP window, enter name and attribute information and click ADD SESSION DATA

7. Go to the browser window that displays the HTTP access log for the server that served the initial request. Click OK to refresh the information.

Notice that even though the server is disabled in `loadbalancer.xml`, it is serving this request.

8. Wait five minutes for the quiescing period to end, then use your open Cluster JSP application window to enter additional name attribute information and click ADD SESSION DATA.

The quiescing period is controlled by the `disable timeout` in minutes for the server instance. Earlier in the tutorial we set this to five minutes.

9. Because the quiescing period has passed, the request is sent to a different application server instance.

To see this result, refresh your server instance's HTTP access log. You should see no new requests.

10. Open a third browser window and view the HTTP access log for the other server instance. You should see the Cluster JSP request being serviced by the second server instance.

You can then shut down the application server instance safely.

You can also quiesce an application to take it offline safely. For more information on application server instance and application quiescing, see the *Sun Java System Application Server Administration Guide*.

Summary and Next Steps

By following the *Getting Started Guide*, you explored these key aspects of the Sun Java System Application Server:

- Key features and architecture
- Using tools for administering the application server
- Deploying and using a J2EE application
- Configuring and using load balancing
- Configuring and using HTTP session persistence
- Quiescing a server instance

Review the next steps in the following table to determine the resources that are best suited to your interests. The left column contains topics of interest and the right column contains information on the next steps to take to pursue your interest.

Table 5-1 Next Steps

Interest	Next Steps
Learn more about Enterprise Edition features implemented in Sun Java System Application Server sample applications.	See the sample applications for Enterprise Edition installed with the Application Server at http://application_server_host_name:port/samples . An index page provides information on all the available Enterprise Edition samples.
Learn about J2EE design and development best practices.	See the Java BluePrints and try out the Java Pet Store and Smart Ticket sample applications that are bundled with the application server.
Learn more about failover, session persistence, and high-availability session configuration in applications.	See the Sun Java System <i>Application Server Application Guidelines for Storing Session State</i> and use the Duke's Bookstore and Session Storage sample applications that are bundled with the application server.

Table 5-1 Next Steps

Interest	Next Steps
Learn about administering Sun Java System Application Server 7, including clustering, session persistence, load balancing, and the HADB.	Review the <i>Sun Java System Application Server 7 Administrator's Guide</i> .
Learn more about J2EE by using tutorials.	See the J2EE Tutorial and Java Web Services Tutorials.
Learn how specific J2EE features are implemented on Sun Java System Application Server 7.	See the Sample Applications included in the application server installation.
Learn about general development practices using Sun Java System Application Server 7.	See the <i>Sun Java System Application Server 7 Developer's Guide</i> .

Index

A

- About Session Persistence Types [22](#)
- access log [65](#)
- Admin Server [16](#)
 - event log [42](#)
 - password [42](#)
 - port [42](#)
 - username [42](#)
 - verifying start-up [41](#)
- Administration interface [27](#)
 - accessing [41](#)
 - viewing logs [63](#)
- administrative domains [18](#)
- application server
 - starting [39](#)
 - verifying start-up [41](#)
- Application Server Instances [16](#)
- application server instances [16](#)
 - adding to cluster [46](#)
 - quiescing [78](#)
 - verifying start-up [43](#)
- applications, sample
 - adding to cluster [66](#)
 - copying [56](#)
 - monitoring [64](#)
 - quiescing [66](#)
 - running [61](#)
 - troubleshooting [65](#)
- apply changes to web server [67](#)
- asadmin utility [27, 59](#)
 - configure-session-persistence command [60](#)
 - create-jdbc-connection-pool command [59](#)

- create-jdbc-resource command [59](#)
- delete-jdbc-connection-pool command [60](#)
- delete-jdbc-resource command [60](#)
- deploy command [59](#)
- list-instances command [44, 61](#)
- PATH variable set-up [39](#)
- start-instance command [59](#)
- stop-instance command [59](#)

C

- cladmin command [27, 57](#)
 - deploy command [58](#)
 - limitations [60](#)
 - location [57](#)
 - log file [59](#)
 - PATH variable set-up [39](#)
 - requirements [60](#)
 - start-instance [61](#)
 - supported asadmin commands [59](#)
 - syntax [58](#)
- cladmin.log file [59](#)
- Client-side Configuration for RMI/IIOP load
 - balancing [32](#)
- clinstance.conf file [57](#)
- clpassword.conf file [58](#)
- clsetup command [27, 37](#)
- Cluster JSP sample application [56](#)
 - running [68](#)
- clusters [18](#)
 - adding applications [66](#)

- creating [46](#)
- deploying applications [57](#)
- example scenario [29](#)
- command-line utilities [27](#)
- configure-session-persistence command [60](#)
- create-jdbc-connection-pool command [59](#)
- create-jdbc-resource command [59](#)

D

- delete-jdbc-connection-pool command [60](#)
- delete-jdbc-resource command [60](#)
- deploy command [59](#)
- deployment
 - sample applications to cluster [57](#)
 - troubleshooting [65](#)
 - verifying [61](#)
- disable-timeout-in-minutes [48](#)
- distributable applications [57](#)
- documentation
 - guide organization [10](#)
- domains, administrative [18](#)

E

- Enabling Session Persistence [22](#)
- error log, web server [52, 75](#)
- event log
 - Admin Server [42](#)
 - application messages in [64](#)
 - application server instance [43](#)
- Example RMI/IIOP Clustering Scenario [31](#)

F

- file session persistence [24](#)
- Forte for Java [12](#)

H

- ha session persistence [23](#)
- HADB [26](#)
- health checker [49, 76](#)
 - interval-in-seconds [50](#)
 - timeout-in-seconds [50](#)
 - URL [50](#)
- highly-available session persistence [23](#)
- HTTP access log [63](#)
- HTTP Clustering Scenario [29](#)
- HTTP listeners [47](#)
 - in RequestExit log file entries [74](#)
- HTTP server
 - listener [44](#)
 - port [44](#)
 - welcome page [45](#)
- http-listener element, server.xml file [44](#)
- HTTPS listeners [47, 53](#)
- HTTPS routing [53](#)

I

- installation requirements [36](#)
- interval-in-seconds [50](#)

L

- list-instances command [44, 61](#)
- load balancer [21, 50](#)
 - dynamic reconfiguration [52](#)
 - health checker [49, 76](#)
 - HTTPS routing [53](#)
 - properties [52](#)
 - response timeout [53](#)
 - sample loadbalancer.xml [53](#)
 - troubleshooting [74](#)
- loadbalancer.xml file
 - creating [45](#)
 - errors in [75](#)
 - example [45](#)

- sample 53
- loadbalancer.xml.example file 45
- logs
 - access 65
 - enabling verbose logging in web server 51
 - error, web server 52
 - event, Admin Server 42
 - event, application server instance 64
 - HTTP access 63
 - monitoring using tail command 43, 64
 - viewing 63
 - web server error 52
- LogVerbose 51

M

- memory session persistence 24
- modified-attribute 25
- modified-session 24
- monitoring sample applications 64
- monitoring, load balancer 50

P

- password, Admin Server 42
- PATH environment variable, setting 39
- persistence frequency
 - web-method 25
- ports
 - Admin Server 42
 - HTTP server 44
 - inaccessible 42
 - web server 68

Q

- quiescing
 - applications 66
 - server instances 78

R

- reload poll interval 52
- response timeout 53
- rpm 14

S

- sample applications
 - adding to cluster 66
 - Cluster JSP 56
 - copying 56
 - deploying to a cluster 57
 - monitoring 64
 - running 61, 68
 - troubleshooting 65
 - verifying deployment 61
 - WAR files 56
- server instances 16
 - adding to cluster 46
 - verifying start-up 43
- Server-side Configuration for RMI/IIOP failover 31
- session 24
- session persistence 22
 - configuring 24
 - file 24
 - ha 23
 - memory 24
 - verifying 77
- Session Persistence Configuration 24
- Session Persistence Types 22
- Setting the Persistence Frequency 25
- Setting the Persistence Scope 24
- SFSB Checkpointing 26
- showrev 14
- Single Sign-on Session Information 25
- single sign-on, availability of session information 25
- start-domain command 39
- starting application server instances 61
- start-instance command 59, 61
- stderr 64
- stdout 64

stop-instance command [59](#)
Sun customer support [13](#)
Sun Java Studio [12](#)
sun-loadbalancer.dtd file [46](#)

error log [52](#)
web server error log [75](#)
web-method [25](#)
 setting persistence frequency to [25](#)
welcome page, HTTP server [45](#)

T

tail command [43](#), [64](#)
time-based [25](#)
timeout-in-seconds [50](#)
troubleshooting
 load balancer [74](#)
 sample applications [65](#)
tutorials
 Cluster JSP sample [56](#)
 enterprise features configuration [39](#)
 preparing to use [35](#)
 steps [37](#)

U

undeploy command [59](#)
username, Admin Server [42](#)

V

verbose logging [51](#)
verifying
 application deployment [61](#)
 server instance start-up [43](#)

W

web server
 Apache [36](#)
 applying changes [67](#)
 configuring logging [50](#)