



Sun Java™ System

# Application Server 7 Administration Guide

---

2004Q2

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

Part No: 817-5445

Copyright © 2004 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 © 2004 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 CONTREFAÇON.

# Contents

<b>List of Figures</b>	<b>23</b>
<b>List of Tables</b>	<b>25</b>
<b>About This Guide</b>	<b>29</b>
Who Should Use This Guide	29
Using the Documentation	30
How This Guide Is Organized	32
Part I: Server Basics and Administering Global Settings	33
Part II: Managing an Individual Server Instance	33
Part III: Managing HTTP Server Features and Virtual Servers	34
Part IV: Configuring Multiple Server Instances	34
Part V: Appendixes	35
Product Line Overview	36
Standard Edition	36
Enterprise Edition	36
Documentation Conventions	37
General Conventions	37
Conventions Referring to Directories	38
Contacting Sun	39
Give Us Feedback	39
Obtain Training	39
Contact Product Support	39
 <b>Part I Server Basics and Administering Global Settings</b>	 <b>41</b>
<b>Chapter 1 Getting Started with Sun Java System Application Server Administration</b>	<b>43</b>
About Sun Java System Application Server	43
Using the Administration Interface	45

Accessing the Administration Interface .....	45
Using Tabs .....	46
Using Buttons .....	47
Accessing Online Help .....	47
Exiting the Administration Interface .....	48
Using the Command-line Interface .....	48
Accessing the Administration Server .....	48
Accessing Sun Java System Application Server Instances .....	49
Using Sun ONE Studio .....	49
About Configuration Files .....	49
Using the License Commands .....	50
 <b>Chapter 2 Setting Administration Server Preferences .....</b>	<b>51</b>
About the Administration Server .....	51
Starting the Administration Server .....	52
Starting Using the startserv Script .....	53
Starting Using the Command-Line Interface .....	53
Starting Using the Services Window (Windows) .....	54
Starting Using the Start Menu (Windows) .....	54
Shutting Down the Administration Server .....	54
Shutting Down Using the Administration Interface .....	55
Shutting Down Using the stopserv Script .....	55
Shutting Down Using the Command-Line Interface .....	56
Shutting Down Using the Services Window (Windows) .....	57
Accessing the Administration Server Settings .....	57
Viewing Administration Server Control Settings .....	57
Applying Changes to the Administration Server .....	58
Editing HTTP Listener Settings for the Administration Server .....	58
Setting SNMP, Logging, and Security Preferences .....	59
 <b>Chapter 3 Configuring Administrative Domains .....</b>	<b>61</b>
About Administrative Domains .....	61
Implementing Administrative Domains .....	62
Directory Structure .....	62
Process/Port Structure .....	62
Configuring Domains .....	62
Creating Domains .....	63
Example: creating a domain in the default location .....	63
Example: creating a domain somewhere other than the default location .....	64
Example: creating a domain for another user (UNIX only) .....	64
User Permissions on UNIX Platforms .....	64
Deleting Domains .....	65

Example: deleting a domain .....	65
Listing Domains .....	65
Example: listing the domains on a local machine .....	66
Example: listing the domains on the local machine using the remote option: .....	66
Starting Domains .....	66
Example: starting the only domain on a machine: .....	66
Stopping Domains .....	66
Example: Stopping all instances in a domain except for the admin server instance. ....	67
Recreating the Domain Registry .....	67

## **Part II Managing an Individual Server Instance ..... 69**

<b>Chapter 4 Using Sun Java System Application Server Instances .....</b>	<b>71</b>
About Sun Java System Application Server Instances .....	72
Starting and Stopping an Application Server Instance .....	73
Using the Start and Stop Buttons in the Administration Interface .....	74
Using the start-instance and stop-instance Commands .....	74
Using the Windows Services (Windows) .....	74
Using the startserv and stopserv Scripts .....	75
Starting the Application Server Instance in Debug Mode .....	76
Setting the Termination Timeout .....	77
Restarting an Application Server Instance Automatically (UNIX) .....	77
About Restarting Automatically .....	78
Restarting Automatically with /etc/inittab (UNIX) .....	78
Restarting Automatically with the System RC Scripts (UNIX) .....	79
Restarting an Application Server Instance Manually (UNIX) .....	79
Restarting the Server Instance Using the Restart Button (UNIX) .....	80
Restarting the Server Instance Using the restart-instance Command (UNIX) .....	80
Restarting the Server Instance Using the restartserv Script (UNIX) .....	80
About the Watchdog .....	81
Adding an Application Server Instance .....	82
Deleting an Application Server Instance .....	82
Applying Changes to an Application Server Instance .....	83
Viewing Application Server Instance Status .....	85
Configuring JVM Settings .....	86
Configuring General Settings .....	86
Configuring Path Settings .....	87
Configuring JVM Options .....	87
Configuring the JVM Profiler .....	88
Configuring JVM Settings Using the Command-Line Interface .....	88
Configuring Logging Setting and Monitoring Settings .....	89

Changing Application Server Instance Advanced Settings .....	90
<b>Chapter 5 Using Logging .....</b>	<b>93</b>
About Logging .....	94
Logging on the UNIX and Windows Platform .....	94
Default Logging in server.log .....	95
Example of server.log .....	95
Changing Default Log File Location .....	96
Logging Using syslog .....	96
Configuring syslog .....	97
To configure syslog: .....	97
Example of syslog messages .....	100
Logging Using the Windows eventlog .....	100
Using Log Levels .....	100
About Log Levels .....	101
Log Levels Used for syslog Configuration .....	103
About Virtual Servers and Logging .....	103
About Loggers .....	104
About Client Side Logging .....	105
Redirecting Application and Server Log Output .....	106
Log File Management .....	106
Internal-daemon Log Rotation .....	107
Scheduler Based Log Rotation .....	108
Rotation Using Solaris logadm Utility .....	109
Rotation Using Solaris “cron” Utility .....	111
About the crontab Entry Format .....	112
Using the Solaris cron Utility to Schedule Execution of logadm .....	113
Configuring Logging Through the Command-line Interface .....	113
Configuring Logging Through the Administration Interface .....	114
Configuring the Log Service .....	115
Configuring Logging for Application Server Components and Subsystems .....	117
To Specify a Log Level .....	117
To Specify a Log File: (Virtual Server) .....	117
To Specify a Transaction Log Location: (Java Transaction Service) .....	118
Configuring the Directives for Error Logging .....	118
Viewing the Access Log File .....	118
Viewing the Event Log File .....	119
Setting Log Preferences .....	119
Running the Log Analyzer .....	120
Viewing Events (Windows) .....	122

<b>Chapter 6 Monitoring Sun Java System Application Server</b>	<b>125</b>
About Monitoring Sun Java System Application Server	125
Statistics	126
SNMP	126
HTTP Server Monitoring	127
Application Components and Subsystems Monitoring	127
Monitoring for Container Subsystems	128
Monitoring for the ORB Service	129
Monitoring for the Transaction Service	129
Administration Server Security Auditing	129
Configuring Administration Server to Use LDAP	130
Setting Up Audit Services Configuration Files	132
Audit Configuration Settings	133
Viewing Security Audit Output	133
Viewing Security Audit Logs on Non-Solaris 9 Platforms	134
Viewing Security Audit Logs Using BSM Auditing on Solaris	134
Quality of Service (QOS)	135
Extracting Monitoring Data Using the CLI	135
list --monitor Command	136
get --monitor Command	137
CLI Name Mapping	138
Petstore Example	139
Monitorable Object Types	141
Monitorable Attribute Names	143
HTTP Server Monitorable Objects	147
Monitorable HTTP Server Elements	147
Monitorable HTTP Server Attributes	148
Administering the Transaction Service Using the CLI	155
Using HTTP Quality of Service	156
Quality of Service Example	156
Configuring Quality of Service (QOS)	157
Required Changes to the obj.conf File	159
Known Limitations to Quality of Service	159
About SNMP	161
Network Management Station (NMS)	162
Management Information Base (MIB) Objects	163
SNMP Messages	167
SNMP Trap Destinations	168
SNMP Agent Community	168
Setting Up SNMP	169
Using a Proxy SNMP Agent	170
Installing the Proxy SNMP Agent	170
Starting the Proxy SNMP Agent	171

Restarting the Native SNMP Daemon .....	171
Installing the SNMP Master Agent .....	172
Enabling and Starting the SNMP Master Agent .....	173
Starting the Master Agent on Another Port .....	174
Manually Configuring the SNMP Master Agent .....	174
Editing the Master Agent CONFIG File .....	174
Defining sysContact and sysLocation Variables .....	175
Configuring the SNMP Subagent .....	176
Starting the SNMP Master Agent .....	176
Manually Starting the SNMP Master Agent .....	176
Starting the SNMP Master Agent Using the Admin Server .....	177
Enabling the Subagent .....	178
 <b>Chapter 7 Configuring J2EE Containers .....</b>	<b>179</b>
About the Web Container .....	179
Understanding the Web Container's Role .....	180
Web Application Configuration .....	181
Virtual Server Attributes .....	181
Web-module Attributes .....	182
Web Application Deployment .....	183
Dynamic Re-deployment and Hot Deployment .....	183
Single Sign-on Facility .....	184
Logging the Web Container .....	184
Configuring Availability in the Web Container (Enterprise Edition) .....	185
About the EJB Container .....	186
Understanding the EJB Container's Role .....	187
Types of Enterprise Java Beans .....	189
About Message-driven Beans .....	191
Configuring the EJB Container .....	192
Performing General EJB Container Configuration .....	192
Configuring EJB Settings .....	194
Configuring MDB Pool Settings .....	196
Configuring Availability in the EJB Container (Enterprise Edition) .....	196
 <b>Chapter 8 Using Transaction Services .....</b>	<b>199</b>
What Is a Transaction? .....	199
Transactions in J2EE .....	200
Transactional Resource Managers .....	201
Databases .....	201
JMS Providers .....	202
J2EE Connectors .....	202
Local and Distributed Transactions .....	203



Container-Managed Transactions .....	205
Transaction Attributes .....	206
Required .....	207
RequiresNew .....	207
Mandatory .....	207
NotSupported .....	208
Supports .....	208
Never .....	208
Attribute Summary .....	208
Setting Transaction Attributes .....	209
Rolling Back a Container-Managed Transaction .....	210
Synchronizing a Session Bean's Instance Variables .....	211
Methods Not Allowed in Container-Managed Transactions .....	212
Bean-Managed Transactions .....	213
Transaction Service Administration .....	213
Administering Transactions Using the Administration Interface .....	214
Administering Transactions Using the Command-Line Interface .....	216
Listing In-Flight Transactions .....	216
Managing Transactions .....	216
Freezing the Transaction Service .....	217
Monitoring Transactions .....	217
 <b>Chapter 9 Configuring Naming and Resources .....</b>	<b>219</b>
About J2EE Naming Services and Resources .....	219
JDBC Datasources .....	220
Java Mail Sessions .....	220
JMS Destinations .....	221
About Java Naming and Directory Interface (JNDI) .....	221
JNDI Architecture .....	221
J2EE Naming Services .....	222
Naming References and Binding Information .....	223
Naming References in J2EE Standard Deployment Descriptor .....	224
Application Environment Entries .....	225
EJB References .....	225
References to Resource Manager Connection Factories .....	226
Resource Environment References .....	227
UserTransaction References .....	228
Initial Naming Context .....	228
COSNaming Service .....	229
JNDI Connection Factories .....	230
To Create a Custom Resource .....	231
To Create an External JNDI Resource .....	233
Accessing External JNDI Repositories .....	233

Mapping Application Resource References .....	234
About URL Connection Factory Resources .....	235
Mapping Application Resource Environment References .....	235
Mapping EJB References .....	236
About Persistence Manager Resources .....	236
What is Persistence? .....	237
Role of the Persistence Manager .....	238
Pre-Deployment Bean Configuration .....	238
Creating a New Persistence Manager .....	240
About JDBC Resources .....	241
About the JDBC API .....	242
What Does The JDBC API Do? .....	242
About Database Access Models .....	243
About JDBC Datasources .....	244
Properties Of a DataSource Object .....	244
Registering a JDBC Resource .....	246
About JDBC Connections .....	247
About JDBC URLs .....	248
Configuring JDBC Connection Pools .....	250
About Connection Pooling .....	258
Monitoring JDBC Connection Pooling .....	260
About Connection Sharing .....	260
About JDBC Transactions .....	261
About Java Mail Resources .....	262
About the JavaMail Message-handling Process .....	263
About the Architectural Components of JavaMail .....	265
Message Class .....	265
Message Storage and Retrieval .....	265
Message Composition and Transport .....	266
About JavaBeans Activation Framework (JAF) .....	266
About JavaMail Configuration Parameters .....	267
J2EE Deployment Descriptor for JavaMail Session References .....	269
Entries in Application Server Deployment Descriptor .....	270
Creating a New JavaMail Session .....	270
Configuring Advanced Resource Properties .....	271
 <b>Chapter 10 Using the JMS Service .....</b>	<b>273</b>
About JMS .....	274
Basic Messaging System Concepts .....	274
Message .....	275
Message Service Architecture .....	275
Message Delivery Models .....	276
JMS Specification .....	276

JMS Message Structure .....	276
JMS Programming Model .....	277
Administered Objects: Provider Independence .....	278
Message-driven Beans .....	279
Built-in JMS Service .....	281
About Sun Java System Message Queue .....	281
Components of an MQ Messaging System .....	282
MQ Message Server .....	283
MQ Client Runtime .....	285
MQ Administered Objects .....	285
MQ Administration Tools .....	286
Integration of MQ with Sun Java System Application Server .....	286
Architecture of the Built-in JMS Service .....	287
Disabling the Built-in JMS Service .....	288
Administration of the Built-in JMS Service .....	290
Configuring the JMS Service .....	291
Managing Physical Destinations .....	293
Create a Physical Destination .....	293
List Physical Destinations .....	294
Delete a Physical Destination .....	294
Managing Administered Object Resources .....	294
Administered Object Attributes .....	295
Administered Object Resource Management Tasks .....	297
Viewing the JMS Log .....	300
Administering the Built-in JMS Service Using the Command-Line Interface .....	301
 <b>Chapter 11 Configuring the Server For Corba/IOP Clients .....</b>	<b>303</b>
About Support for CORBA/IOP Clients .....	303
About Interoperability .....	304
About the ORB .....	304
About the RMI/IOP Functionality .....	305
About the Authentication Process .....	305
Configuring the ORB .....	306
To Perform General ORB Configuration .....	306
To Configure IOP Listener For the ORB .....	308
 <b>Chapter 12 Configuring the Web Server Plug-in .....</b>	<b>311</b>
About the Web Server Plug-in .....	312
Handling Client Requests .....	312
HTTP Basics .....	312
Steps in the Request Handling Process .....	314
Web Server Plug-in Configuration .....	315

Web Server Plug-in SAF Reference .....	316
init-passthrough .....	316
auth-passthrough .....	316
service-passthrough .....	317
check-passthrough .....	318
Using The Web Server Plug-in .....	319
Configuring Sun Java System Web Server .....	319
Changes to Sun Java System Web Server .....	319
Changes to Sun Java System Application Server .....	321
Configuring Application Server as a Proxy .....	322
Configuring Microsoft IIS .....	324
Configuring the Web Server Plug-in for IIS .....	324
Configuring IIS to Use the Web Server Plug-in .....	325
Configuring Multiple Server Pools .....	326
Sample sun-passthrough.properties File .....	327
Configuring Apache Web Server .....	329
Minimum Requirements .....	330
Compiling Apache With mod_proxy Module .....	330
Modifying the httpd.conf File .....	332
Starting And Stopping Apache .....	332
 <b>Chapter 13 Deploying Applications .....</b>	<b>333</b>
About J2EE Modules .....	334
About J2EE Applications .....	335
J2EE Standard Descriptors .....	335
Application Server Descriptors .....	336
Naming Standards .....	337
Deployment Directory Structure .....	338
Runtime Environments .....	340
Module Runtime Environment .....	340
Application Runtime Environment .....	341
About Classloaders .....	342
Deploying Modules and Applications .....	342
Deployment Names and Errors .....	343
The Deployment Life Cycle .....	343
Dynamic Deployment .....	343
Disabling a Deployed Application or Module .....	344
Dynamic Reloading .....	344
Using Availability Features (Enterprise Edition) .....	345
Tools for Deployment .....	345
The asadmin Utility .....	346
The Administration Interface .....	347
Sun ONE Studio .....	348

Deploying a Module or Application .....	348
Deploying a WAR Module .....	348
Deploying an EJB JAR Module .....	349
Deploying a Lifecycle Module .....	349
The asadmin Utility .....	349
The Administration Interface .....	350
Deploying an RMI/IIOP Client .....	351
Deploying a J2EE CA Resource Adapter .....	352
Deploying Static Content .....	352
Access to Shared Frameworks .....	352
The Application Deployment Descriptor Files .....	352

## **Part III Managing HTTP Server Features and Virtual Servers ..... 353**

<b>Chapter 14 Configuring HTTP Features .....</b>	<b>355</b>
About the HTTP Features .....	355
Configuring the File Cache .....	356
Tuning Your Server for Performance .....	356
Configuring HTTP Quality of Service .....	357
Adding and Using Thread Pools .....	359
Editing Advanced Settings .....	359
Configuring MIME Types .....	360

<b>Chapter 15 Using Virtual Servers .....</b>	<b>363</b>
Virtual Servers Overview .....	363
HTTP Listeners .....	364
Virtual Servers .....	365
Types of Virtual Servers .....	366
IP-Address-Based Virtual Servers .....	366
URL-Host-Based Virtual Servers .....	366
Default Virtual Server .....	367
obj.conf File .....	367
Virtual Server Selection for Request Processing .....	368
Document Root .....	368
Using Sun Java System Application Server Features with Virtual Servers .....	369
Using SSL with Virtual Servers .....	369
Using Access Log Files and Server Log Files .....	370
Using Access Control with Virtual Servers .....	370
Using CGIs with Virtual Servers .....	371
Creating and Configuring HTTP Listeners .....	371
Creating an HTTP Listener .....	371

Editing HTTP Listener Settings .....	372
Deleting an HTTP Listener .....	373
Creating and Configuring Virtual Servers .....	374
Creating a Virtual Server .....	374
Required Settings .....	375
Optional General Settings .....	375
Web Application Settings .....	376
CGI Settings .....	377
HTTP Quality of Service Settings .....	377
Editing Virtual Server Settings .....	378
Editing General Settings Using the Administration Interface .....	378
Editing General Settings Using the Command-Line Interface .....	379
Editing CGI Settings .....	380
Editing Document Handling Settings, Document Directories Settings, and HTTP/HTML Settings .....	380
Deleting a Virtual Server .....	380
Deploying Virtual Servers .....	381
Example 1: Default Configuration .....	381
Example 2: Secure Server .....	383
Example 3: Intranet Hosting .....	384
Example 4: Mass Hosting .....	386
 <b>Chapter 16 Managing Virtual Server Content .....</b>	<b>389</b>
Changing the Document Root .....	390
Setting Additional Document Directories .....	390
Enabling Remote File Manipulation .....	391
Using htaccess .....	392
Restricting Symbolic Links (UNIX) .....	392
Customizing User Public Information Directories (UNIX) .....	393
Configuring Public Information Directories .....	394
Restricting Content Publication .....	395
Loading the Entire Password File on Startup .....	395
Setting the Document Preferences .....	395
Entering an Index Filename .....	396
Selecting Directory Indexing .....	396
Specifying a Server Home Page .....	397
Specifying a Default MIME Type .....	397
Customizing Error Responses .....	397
Changing the International Character Set .....	398
Setting the Document Footer .....	400
Configuring URL Forwarding .....	401
Setting up Server-Parsed HTML .....	401
Setting Cache Control Directives .....	402

Using Stronger Ciphers .....	403
------------------------------	-----

## Part IV Configuring Multiple Server Instances ..... 405

### Chapter 17 Configuring HTTP Load Balancing and Failover (Enterprise Edition) ..... 407

About Load Balancing and Failover .....	407
---	-----

Requirements for Load Balancing .....	408
---------------------------------------	-----

Using Third Party Load Balancers .....	408
--	-----

Understanding Assigned Requests and Unassigned Requests .....	409
---	-----

HTTP Load Balancing Algorithm .....	409
-------------------------------------	-----

About Sticky Round Robin Load Balancing Algorithm .....	409
---	-----

Configuring HTTP Load Balancing and Failover .....	410
--	-----

Configuration Changes to Web Server .....	410
---	-----

Modifications to Sun Java System Web Server .....	411
---	-----

Modifications to Apache Web Server .....	412
--	-----

Creating Load Balancer Configuration File .....	413
---	-----

Configuring Multiple Web Server Instances .....	416
---	-----

Configuring Multiple Sun Java System Web Server Instances .....	417
---	-----

Configuring Multiple Apache Web Server Instances .....	418
--	-----

Administering HTTP Load Balancing and Failover .....	418
--	-----

Enabling/Disabling Application Server Instances .....	419
---	-----

Restarting Application Server Instances .....	420
---	-----

Enabling Dynamic Reconfiguration .....	420
--	-----

Configuring HTTP/S Session Failover .....	422
---	-----

About HTTPS Routing .....	422
---------------------------	-----

Configuring HTTPS Routing .....	422
---------------------------------	-----

Configuring Response Timeout .....	423
------------------------------------	-----

Configuring Quiescence .....	423
------------------------------	-----

Quiescing an Instance .....	424
-----------------------------	-----

Quiescing an Application .....	424
--------------------------------	-----

Configuring The Health-Checker .....	425
--------------------------------------	-----

Configuring Idempotent URL Requests .....	426
---	-----

Specifying an Idempotent URL .....	426
------------------------------------	-----

Characteristics of an Idempotent URL .....	427
--	-----

Configuring HTML Error Pages .....	427
------------------------------------	-----

Rules Applicable for Displaying Error Pages .....	428
---	-----

Monitoring the Load Balancer Plug-in .....	429
--	-----

Configuring Log Messages .....	429
--------------------------------	-----

Types of Log Messages .....	429
-----------------------------	-----

Configurator Error Messages .....	430
-----------------------------------	-----

Configuring Monitoring .....	431
------------------------------	-----

Understanding Monitoring Messages .....	431
Known Issues in Load Balancing Requests .....	432
<b>Chapter 18 Configuring RMI/IIOP Load Balancing and Failover (Enterprise Edition) ....</b>	<b>435</b>
About RMI/IIOP Load Balancing and Failover .....	435
Requirements for Load Balancing and Failover .....	436
About RMI/IIOP Load Balancing and Failover Algorithm .....	436
Initial Context Based Failover Algorithm .....	437
Configuring RMI/IIOP Failover .....	437
Using Admin Console to Configure RMI/IIOP Load Balancing .....	438
Using CLI to Configure RMI/IIOP Load Balancing .....	440
Sample server.xml Configuration .....	440
Sample availability-service Configuration in server.xml .....	440
<b>Chapter 19 Managing Clusters (Enterprise Edition) .....</b>	<b>443</b>
About Application Server Clusters .....	443
Defining a Cluster .....	445
Requirements for Adding an Application Server Instance to a Cluster .....	447
Adding Application Server Instances to a Cluster .....	447
Defining Multiple Clusters .....	449
Administering Application Server Clusters .....	449
Starting a Cluster .....	450
Deploying a Web Application .....	450
Undeploying a Web Application .....	452
Redeploying an Existing Web Application on a Cluster .....	452
Stopping an Application Server Instance in a Cluster .....	453
Removing an Application Server Instance from a Cluster .....	453
Stopping a Cluster .....	454
Removing a Cluster .....	454
Sample Cluster Configuration in a loadbalancer.xml File .....	455
Online Reconfiguration of HTTP Clusters .....	455
Disabling and Quiescing an Application Server Instance in a Cluster .....	456
Disabling and Quiescing a Web Application in a Cluster .....	458
Modifying the Quiescing Period While Quiescing is On .....	459
Using Multiple Clusters for Online Upgrades Without Loss of Service .....	460
Reconfiguring an Application Server Instance in a Running Cluster .....	461
Using Multiple Load Balancers .....	461
<b>Chapter 20 Configuring Session Persistence (Enterprise Edition) .....</b>	<b>463</b>
About Session Persistence .....	463
Basic Session Persistence Configuration Steps .....	464
Setting Up Availability .....	466



Configuring JDBC Parameters for the HADB Database .....	466
Referencing the HADB Database's JDBC Resource .....	467
Creating the Session Store .....	468
Clearing the Session Store .....	470
Before You Clear the Session Store .....	471
Using the <code>asadmin clear-session-store</code> Command .....	471
After You Clear the Session Store .....	472
Managing the Session Store for Clusters .....	472
Enabling and Disabling Availability .....	474
Server Instance and Container Levels .....	475
Application and Module Levels .....	476
The SFSB Level .....	478
Failover of J2EE Object References Stored in Sessions .....	478
Configuring SFSB Session Persistence .....	479
How SFSB Session Persistence Works When Availability Is Enabled .....	480
Setting the SFSB Session Store When Availability is Disabled .....	480
Configuring HTTP Session Persistence .....	481
Making a Web Application Distributable .....	481
Availability of Single Sign-on in the HTTP Session State .....	481
Session Availability for Applications Belonging to a Single Sign-On Group .....	481
Setting the Session Timeout .....	482
HTTP Session Persistence Options .....	483
About the <code>asadmin configure-session-persistence</code> Command .....	484
Using the <code>cladmin</code> Command to Configure Session Persistence for an Entire Cluster .....	485
Setting the Persistence Type .....	485
Comparison of Persistence Type Options .....	492
Setting the Persistence Frequency .....	492
Comparison of Persistence Frequency Options .....	496
Setting the Persistence Scope .....	496
Comparison of Persistence Scope Options .....	501
Configuring Other HTTP Session Persistence Properties .....	502
Overriding HTTP Session Persistence Settings .....	504
 <b>Chapter 21 Administering the High-Availability Database (Enterprise Edition) .....</b>	<b>507</b>
About the High-Availability Database .....	508
HADB Architecture .....	509
HADB Nodes .....	510
The <code>hadbm</code> Command .....	510
Creating Other Management Clients .....	512
Configuring the HADB .....	514
Setting the Management Protocol .....	514
Creating a Database .....	515
Setting Heterogeneous Device Paths .....	519

Configuring Double Networks .....	520
Additional Steps for inetd .....	520
Setting Up the JDBC Connection Pool .....	522
Getting the JDBC URL .....	522
Creating a Connection Pool .....	523
Connection Pool Example .....	524
Creating a JDBC Resource .....	525
Managing the HADB .....	525
Starting a Node .....	526
Stopping a Node .....	527
Restarting a Node .....	528
Starting the HADB .....	529
Stopping the HADB .....	529
Restarting the HADB .....	530
Listing Databases .....	531
Clearing the HADB .....	531
Removing a Database .....	532
Expanding the HADB .....	533
Adding Storage Space to Existing Nodes .....	533
Adding Machines .....	534
Adding Nodes to the HADB .....	534
Refragmenting the HADB .....	536
Adding Nodes Without Refragmenting .....	537
Monitoring the HADB .....	539
Getting the Status of the HADB .....	539
Database Status .....	540
Node Status .....	540
Getting Device Information .....	542
Maintaining the HADB Machines .....	544
Viewing and Modifying Configuration Attributes .....	546
Getting the Values of Configuration Attributes .....	546
Setting the Values of Configuration Attributes .....	547
Configuration Attributes .....	548
Clearing and Archiving History Files .....	553
Recovering from Session Data Corruption .....	554
Using Sun Customer Support for the HADB .....	554
Environment Variables .....	555

**Part V Appendixes ..... 557**

<b>Appendix A Using the Command Line Interface .....</b>	<b>559</b>
About the Command Line Interface .....	559
About the asadmin Utility .....	560
About Ant Tasks .....	560
About Other Command Line Utilities .....	560
Using asadmin .....	561
Understanding the Command Syntax .....	562
Command .....	562
Options .....	562
Boolean Options .....	562
Operands .....	563
Syntax Example .....	563
Using Singlemode and Multimode .....	563
Singlemode .....	564
Multimode .....	564
Multiple Multimode .....	564
Using Interactive and Non-Interactive Options .....	565
Using the Environment Commands .....	565
Using the Password File Option .....	567
Running asadmin Locally or Remotely .....	568
Using Command Line Invocations .....	569
Using asadmin from the Command Line .....	569
Using asadmin with Input from a File (Script) .....	570
Using asadmin with Standard Input (Pipe) .....	570
Using Escape Characters .....	570
Escape Characters on UNIX in Singlemode .....	571
Escape Characters on Windows in Singlemode .....	571
Escape Characters on All Platforms in Singlemode .....	572
Escape Characters on All Platforms in Multimode .....	572
Using get and set Commands .....	572
get and set Command Examples .....	574
Getting and Setting Multiple Values Examples .....	575
Monitoring Using get and set Commands .....	575
Using Help .....	575
Viewing Output and Errors .....	576
Viewing the Exit Status .....	576
Viewing Usage .....	578
Security Considerations .....	578
Concurrent Access Considerations .....	578
Command Reference .....	579
List of Commands .....	579

List of Dotted Names and Attributes .....	584
Dotted Names Used in asadmin .....	584
Service Names .....	584
Resource Names .....	585
Application Names .....	585
Other Names .....	586
Attributes .....	586
jms-service .....	587
transaction-service .....	587
mdb-container .....	588
ejb-container .....	589
web-container .....	590
java-config .....	590
orb or iiop-service .....	591
orblister or iiop-listener .....	592
log-service .....	593
security-service .....	594
http-service .....	595
jdbc-resource .....	595
jndi-resource .....	596
jdbc-connection-pool .....	597
custom-resource .....	598
jms-resource .....	598
persistence-manager-factory-resource .....	599
mail-resource .....	599
application .....	600
ejb-module .....	601
web-module .....	602
connector-module .....	603
http-listener or http-server.http-listener .....	604
mime .....	605
acl .....	605
virtual-server .....	606
auth-db .....	607
authrealm .....	608
lifecycle-module .....	608
profiler .....	609
server configuration (name of server instance) .....	610
Long and Short Option Formats, Default Values, and Environment Variable Equivalents ....	611
 <b>Appendix B Failover Scenarios (Enterprise Edition) .....</b>	<b>615</b>
HTTP Session Failover .....	615
Assumptions and Requirements .....	616

Persistence Type Set to memory for Application Server Instances in Cluster .....	617
Persistence Type Set to memory for an Application Server Instance .....	617
Persistence Type Set to file for an Application Server Instance .....	618
Persistence Type Set to ha for Application Server Instances in a Cluster .....	619
Persistence Type Set to memory for a Web Application .....	621
Persistence Type Set to file for a Web Application .....	621
Persistence Type Set to ha for a Web Application .....	622
Stateful Session Bean Failover .....	623
Assumptions and Requirements .....	623
SFSB State Stored in the File System .....	624
SFSB State Stored in the HADB .....	625
Failover on RMI/IOP Path .....	626
Assumptions and Requirements .....	627
RMI/IOP Failover Scenario .....	627
 <b>Appendix C Compiling and Configuring Apache Web Server (Enterprise Edition) .....</b>	<b>629</b>
Minimum Requirements .....	629
Installing SSL-aware Apache .....	630
Compiling and Building Open SSL .....	631
Configuring Apache with mod SSL .....	631
Compiling and Building Apache .....	631
Starting and Stopping Apache .....	632
 <b>Appendix D Running Message Queue Broker with Trial Enterprise License .....</b>	<b>635</b>
About Enterprise Edition .....	635
Running Message Queue Broker with Trial License of Enterprise Edition .....	636
Purchasing the Enterprise Edition License .....	636
 <b>Appendix E HTTP Load Balancer Configuration File .....</b>	<b>637</b>
sun-loadbalancer_1_1.dtd File .....	637
Subelements .....	638
Data .....	639
Attributes .....	639
Elements in the loadbalancer.xml File .....	639
 <b>Appendix F Using the cladmin Command for Administration (Enterprise Edition) .....</b>	<b>647</b>
About the cladmin Command .....	647
Cluster Operations Using cladmin Command .....	648
asadmin Commands Supported by cladmin .....	648
Requirements for Running the cladmin Command .....	649
Using cladmin Command .....	650
Input Files for the cladmin Command .....	650

clinstance.conf File .....	650
clpassword.conf File .....	652
Running the cladmin Command .....	653
cladmin Status Messages .....	654
Log file for the cladmin Command .....	655
Limitations of cladmin Command .....	655
 <b>Glossary .....</b>	 <b>657</b>
 <b>Index .....</b>	 <b>687</b>

# List of Figures

Figure 1-1	Closed Symbol .....	46
Figure 1-2	Open Symbol .....	46
Figure 2-1	Warning Icon .....	58
Figure 4-1	Warning Icon .....	84
Figure 7-1	Web Container within the Application Server Architecture .....	180
Figure 8-1	J2EE Components Accessing Resources in a Transaction .....	204
Figure 8-2	Transaction Attributes .....	206
Figure 9-1	Overview of the JNDI Architecture .....	222
Figure 9-2	Basic Persistence Scheme .....	237
Figure 9-3	Message Handling Process of the Java Mail API .....	264
Figure 10-1	Message Service Architecture .....	275
Figure 10-2	JMS Programming Objects .....	277
Figure 10-3	MDB Message Consumers .....	280
Figure 10-4	MQ System Architecture .....	283
Figure 10-5	Messaging Operations .....	285
Figure 10-6	Built-in MQ Messaging System .....	287
Figure 12-1	Relationship Between Web Browser, Web Server, Application Server, and Web Server Plug-in SAFs .....	315
Figure 13-1	Module Runtime Environment .....	340
Figure 13-2	Application Runtime Environment .....	341
Figure 15-1	Default Configuration .....	382
Figure 15-2	Secure Server .....	383
Figure 15-3	Intranet Hosting Using URL-Host-Based Virtual Servers .....	384
Figure 15-4	Intranet Hosting Using IP-Address-Based Virtual Servers .....	386
Figure 15-5	Mass Hosting .....	387
Figure 21-1	HADB Architecture .....	509





# List of Tables

Table 1	Sun Java System Application Server Documentation Roadmap .....	30
Table 1-1	Standard Administration interface Buttons .....	47
Table 5-1	Log Levels .....	102
Table 5-2	Sun Java System Application Server Log Levels Mapped to server.log .....	103
Table 5-3	Subsystems and Locations in Sun Java System Application Server .....	104
Table 5-4	ACC Logging Element .....	106
Table 5-5	Log Service Attributes .....	116
Table 6-1	http-server .....	143
Table 6-2	virtual-server .....	143
Table 6-3	process .....	143
Table 6-4	orb-connection .....	144
Table 6-5	orb-thread-pool .....	144
Table 6-6	jdbc-connection-pool .....	144
Table 6-7	transaction-service .....	144
Table 6-8	bean-pool .....	145
Table 6-9	bean-cache .....	145
Table 6-10	bean-method .....	146
Table 6-11	Monitorable HTTP Server Elements .....	147
Table 6-12	Server .....	148
Table 6-13	ConnectionQueue .....	149
Table 6-14	ThreadPool .....	149
Table 6-15	Profile .....	149
Table 6-16	Process .....	150
Table 6-17	ConnectionQueueBucket .....	150
Table 6-18	ThreadPoolBucket .....	151
Table 6-19	DnsBucket .....	151
Table 6-20	KeepaliveBucket .....	152

Table 6-21	CacheBucket .....	152
Table 6-22	Thread .....	153
Table 6-23	VirtualServer .....	153
Table 6-24	RequestBucket .....	154
Table 6-25	ProfileBucket .....	155
Table 6-26	appserv.mib Managed Objects and Descriptions .....	164
Table 6-27	Procedure for Setting Up SNMP .....	169
Table 7-1	Monitoring Statistics for EJB Container .....	193
Table 8-1	Transaction Attributes .....	209
Table 8-2	Monitorable Attributes of Java Transaction Services .....	214
Table 9-1	JNDI Lookups and Their Associated References .....	224
Table 9-2	General Settings .....	251
Table 9-3	Connection Pool Settings .....	252
Table 9-4	Connection Validation and Transaction Isolation .....	252
Table 9-5	Options required for creating JDBC Connection Pools using the Command Line Interface .....	254
Table 9-6	JDBC Connection Pool Parameters for Monitoring .....	260
Table 9-7	JavaMail API MIME Data Type to Java Type Mappings .....	267
Table 10-1	Comparison of Sun Java System Message Queue and Sun Java System Application Server Administration Capability .....	290
Table 10-2	JMS Service Properties .....	291
Table 10-3	Connection Factory Administered Object Attributes .....	295
Table 10-4	asadmin Commands for Administering the Built-in JMS Service .....	301
Table 13-1	J2EE Standard Descriptors .....	336
Table 13-2	Sun Java System Application Server Descriptors .....	337
Table 20-1	Environment Variables for the create-session-store Command Options .....	469
Table 20-2	Environment Variables for the create-session-store Command Options .....	471
Table 20-3	Object Types Supported for Session State Failover .....	478
Table 20-4	Comparison of Persistence Type Options for HTTP Sessions .....	492
Table 20-5	Comparison of Persistence Frequency Options .....	496
Table 20-6	Comparison of Persistence Scope Options .....	501
Table 20-7	manager-properties Configurable for Session Persistence .....	503
Table 20-8	store-properties Configurable for Session Persistence .....	503
Table 20-9	session-properties Configurable for Session Persistence .....	504
Table 21-1	HADB Roadmap .....	507
Table 21-2	hadbm General Options .....	511
Table 21-3	hadbm create Options .....	516
Table 21-4	HADB Connection Pool Settings .....	523

Table 21-5	HADB Connection Pool Properties .....	523
Table 21-6	HADB JDBC Resource Settings .....	525
Table 21-7	hadbm startnode Options .....	526
Table 21-8	hadbm stopnode Options .....	527
Table 21-9	hadbm restartnode Options .....	528
Table 21-10	hadbm clear Options .....	532
Table 21-11	hadbm addnodes Options .....	535
Table 21-12	hadbm refragment Options .....	537
Table 21-13	hadbm status Options .....	540
Table 21-14	Configuration Attributes .....	548
Table 21-15	HADB Options and Environment Variables .....	555
Table A-1	Other Command-Line Utilities .....	560
Table A-2	Environment Variables Used with export Command .....	567
Table A-3	get and set Commands .....	573
Table A-4	asadmin Commands .....	579
Table A-5	Service Names for Command-Line Interface .....	584
Table A-6	Resource Names for Command-Line Interface .....	585
Table A-7	Application Names for Command-Line Interface .....	586
Table A-8	Other Item Names for Command-Line Interface .....	586
Table A-9	JMS Service Attributes .....	587
Table A-10	Transaction Service Attributes .....	588
Table A-11	MDB Container Attributes .....	588
Table A-12	EJB Container Attributes .....	589
Table A-13	Web Container Attributes .....	590
Table A-14	JVM Attributes .....	590
Table A-15	ORB/IIOP Service Attributes .....	591
Table A-16	IIOP Listener Attributes .....	592
Table A-17	Log Configuration Attributes .....	593
Table A-18	Security Realm Configuration Attributes .....	594
Table A-19	HTTP Service Attributes .....	595
Table A-20	JDBC Resource Attributes .....	596
Table A-21	JNDI Resource Attributes .....	596
Table A-22	JDBC Connection Pool Attributes .....	597
Table A-23	Custom Resource Attributes .....	598
Table A-24	JMS Resource Attributes .....	598
Table A-25	Persistence Manager Factory Resource Attributes .....	599
Table A-26	Java Mail Resource Attributes .....	600

Table A-27	Application Attributes .....	600
Table A-28	EJB Module Attributes .....	601
Table A-29	WEB Module Attributes .....	602
Table A-30	Connector Module Attributes .....	603
Table A-31	HTTP Listener Attributes .....	604
Table A-32	MIME Types Attributes .....	605
Table A-33	ACL Attributes .....	606
Table A-34	Virtual Server Attributes .....	606
Table A-35	Authentication Database Attributes .....	607
Table A-36	Authorization Realm Attributes .....	608
Table A-37	LifeCycle Module Attributes .....	609
Table A-38	JVM Profiler Configuration Attributes .....	609
Table A-39	Server Configuration Attributes .....	610
Table A-40	Short and Long Options, Default Values, and Environment Variables .....	611
Table E-1	requirement rules and subelement suffixes .....	638
Table E-2	loadbalancer subelements .....	640
Table E-3	property subelements .....	640
Table E-4	property attributes .....	641
Table E-5	Load Balancer Properties .....	641
Table E-6	cluster subelements .....	642
Table E-7	cluster attributes .....	642
Table E-8	instance attributes .....	643
Table E-9	web-module subelements .....	643
Table E-10	web-module attributes .....	644
Table E-11	idempotent-url-pattern attributes .....	644
Table E-12	Health checker properties .....	645
Table F-1	asadmin Commands Supported for Use with cladmin Command .....	649
Table F-2	Entries in the clinstance.conf File .....	651
Table F-3	Exit Codes for the cladmin Command .....	655

# About This Guide

This guide describes how to configure and administer Sun Java™ System Application Server Standard and Enterprise Edition 7 2004Q2.

It is intended for information technology administrators in the corporate enterprise who want to extend client-server applications to a broader audience through the World Wide Web.

This preface addresses the following topics:

- [Who Should Use This Guide](#)
- [Using the Documentation](#)
- [How This Guide Is Organized](#)
- [Product Line Overview](#)
- [Documentation Conventions](#)
- [Contacting Sun](#)

## Who Should Use This Guide

The intended audience for this guide is the person who configures, administers, and deploys the Application Server subsystems and components.

This guide assumes you are familiar with the following topics:

- Java programming and concepts
- Client/server programming model
- Installation of enterprise-level software products
- Familiarity in managing large enterprise systems.

# 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 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*)

<b>For information about</b>	<b>See the following</b>
( <i>Updated for 7 2004Q2</i> ) 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>
( <i>Updated for 7 2004Q2</i> ) 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>
( <i>Updated for 7 2004Q2</i> ) 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>
( <i>Updated for 7 2004Q2</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>
( <i>Updated for 7 2004Q2</i> ) 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>
( <i>Updated for 7 2004Q2</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>
( <i>Updated for 7 2004Q2</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>
( <i>Updated for 7 2004Q2</i> ) 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: <a href="http://docs.sun.com/db?p=prod/s1.s1msgqu">http://docs.sun.com/db?p=prod/s1.s1msgqu</a>

# How This Guide Is Organized

This guide is divided into four parts, plus a comprehensive index.

Begin with [Part I, “Server Basics and Administering Global Settings,”](#) for an overview of the product.

[Part II, “Managing an Individual Server Instance”](#) introduces you to using the Administration Server, and to using other server functions that affect all server instances.

Once you are familiar with the fundamentals of using the Administration Server, you can refer to [Part III, “Managing HTTP Server Features and Virtual Servers,”](#) which provides information for using programs and configuration styles.

For configuring multiple application server instances, setting up clustering, configuring HTTP session load balancing and failover, refer to [Part IV, “Configuring Multiple Server Instances.”](#)

Finally, [Part V, “Appendixes”](#) addresses specific reference topics that describe various topics, including internationalization issues, server extensions, failover scenarios, and the Sun Java System Application Server command line interface documentation.



## Part I: Server Basics and Administering Global Settings

This part provides an overview of the Application Server. The following chapters are included:

- [Chapter 1, “Getting Started with Sun Java System Application Server Administration,”](#) provides an overview of Application Server.
- [Chapter 2, “Setting Administration Server Preferences,”](#) describes how to manage your Administration Server.
- [Chapter 3, “Configuring Administrative Domains,”](#) describes how to use multiple domains.

## Part II: Managing an Individual Server Instance

This part provides conceptual and procedural details about configuring, managing, and using server instances. The following chapters are included:

- [Chapter 4, “Using Sun Java System Application Server Instances,”](#) describes how to configure server preferences for your Application Server.
- [Chapter 5, “Using Logging,”](#) describes the foundation for logging, and the logging features and functions within Application Server.
- [Chapter 6, “Monitoring Sun Java System Application Server,”](#) contains information about the monitoring and Simple Network Management Protocol (SNMP) features and functions available within Application Server.
- [Chapter 7, “Configuring J2EE Containers,”](#) explains how to configure and use the container that provide runtime support for J2EE application components such as Enterprise Java Beans (EJBs) and Message Driven Beans (MDBs).
- [Chapter 8, “Using Transaction Services,”](#) explains database transactions and how to use manage them.
- [Chapter 9, “Configuring Naming and Resources,”](#) explains how to configure J2EE resources.
- [Chapter 10, “Using the JMS Service,”](#) provides information needed to understand and administer the built-in JMS Service provided through Sun Java System Message Queue, the native JMS provider.

- [Chapter 11, “Configuring the Server For Corba/IIOP Clients,”](#) explains how to configure support for CORBA-based clients, using the RMI/IIOP protocol within an Application Server environment.
- [Chapter 12, “Configuring the Web Server Plug-in,”](#) explains how Sun Java System Application Server Standard Edition 7 processes HyperText Transfer Protocol (HTTP) requests, how to configure and use the web server plug-in with the Application Server.
- [Chapter 13, “Deploying Applications,”](#) describes how to deploy applications to the Application Server.

## Part III: Managing HTTP Server Features and Virtual Servers

This part provides information for using the Administration interface to programs and configuration styles. The following chapters are included:

- [Chapter 14, “Configuring HTTP Features,”](#) describes how to configure preferences for your HTTP-related features of your Application Server.
- [Chapter 15, “Using Virtual Servers,”](#) describes how to set up and administer virtual servers using your Application Server.
- [Chapter 16, “Managing Virtual Server Content,”](#) describes how you can configure and manage your server’s content.

## Part IV: Configuring Multiple Server Instances

This section provides information for setting up and configuring clusters, load balancing, session persistence, and the Highly Available database. The following chapters are included:

- [Chapter 17, “Configuring HTTP Load Balancing and Failover \(Enterprise Edition\),”](#) describes how to setup load balancing of HTTP requests, configure, and administer the load balancer plug-in.
- [Chapter 18, “Configuring RMI/IIOP Load Balancing and Failover \(Enterprise Edition\),”](#) describes how to setup load balancing and failover of RMI-IIOP requests

- [Chapter 19, “Managing Clusters \(Enterprise Edition\),”](#) describes how to set up and manage application server clusters. It also provides important information about clusters that will help you fully utilize the failover capabilities of Application Server 7, Enterprise Edition.
- [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\),”](#) describes how to configure session persistence for the Application Server 7, Enterprise Edition.
- [Chapter 21, “Administering the High-Availability Database \(Enterprise Edition\),”](#) describes the High Availability Database (HADB) in the Application Server 7, Enterprise Edition and explains how to configure and administer HADB.

## Part V: Appendixes

This section includes various appendixes with reference material that you may wish to review. This section includes the following appendixes:

- [Appendix A, “Using the Command Line Interface,”](#) provides instructions for using command line utilities in place of the user interface screens.
- [Appendix B, “Failover Scenarios \(Enterprise Edition\),”](#) describes how session persistence settings for application server instances and web applications affect the failover of session information.
- [Appendix C, “Compiling and Configuring Apache Web Server \(Enterprise Edition\),”](#) explains how to compile and build Apache Web Server to work with Application Server.
- [Appendix D, “Running Message Queue Broker with Trial Enterprise License,”](#) describes how to run the Sun Java System Message Queue (MQ) broker using a trial enterprise license so that you can evaluate enterprise features of the MQ broker.
- [Appendix E, “HTTP Load Balancer Configuration File,”](#) describes `loadbalancer.xml` and `sun-loadbalancer_1_1.dtd` files
- [Appendix F, “Using the cladmin Command for Administration \(Enterprise Edition\),”](#) describes the use of the `cladmin` script that is bundled with Sun Java System Application Server.

# Product Line Overview

Sun Java System Application Server is a breakthrough product that raises the bar in application server technologies. It incorporates the latest Java technologies in an easy-to-use, developer-friendly package. The Sun Java System Application Server product leverages over six years of Sun expertise in delivering highly scalable application server technology, enabling developers to rapidly build robust applications that are based on JavaServer Pages™ (JSP™) technology, Java™ Servlet, and Enterprise JavaBeans™ (EJB™) technology. This technology supports a broad range of business requirements from small departmental applications to enterprise-scale, mission-critical services. Two editions of the application server are offered to suit a variety of needs:

- [Standard Edition](#)
- [Enterprise Edition](#)

## Standard Edition

The Standard Edition layers enhanced remote-management capabilities which allow the management of multiple application server instances from a central administration station. This edition also includes the ability to distribute web application traffic through a web server tier proxy. Standard Edition supports configuration of multiple application server instances per administrative domain. Additionally, you can use the Simple Network Monitoring Protocol (SNMP) to monitor your Standard Edition application server.

## Enterprise Edition

Enterprise Edition enhances the core application server platform with high availability, load balancing, and cluster management capabilities suited for the most demanding J2EE-based application deployments. The management capabilities of the Standard Edition are extended in Enterprise Edition to account for multiple-instance deployments.

Clustering support includes groups of cloned application server instances to which client requests can be load balanced. Both the web tier Load Balancing Plug-in and third-party hardware load balancers are supported by this edition. Session failover for various application server components are included in the Enterprise Edition. The patented “Always On” highly available database technology forms the basis for the high availability persistence store in the Enterprise Edition.

For more product information, see the Sun Java System Application Server page on the Sun Microsystems web site, <http://www.sun.com>. Also review the Supplemental Terms included in the product (accepted during installation or download) to understand your rights for each edition of the application server.

## 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 38](#).

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 38](#) 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.

# 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 documentation, please use the following URL:

<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/](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



# Server Basics and Administering Global Settings

Chapter 1, “Getting Started with Sun Java System Application  
Server Administration”

Chapter 2, “Setting Administration Server Preferences”

Chapter 3, “Configuring Administrative Domains”



# Getting Started with Sun Java System Application Server Administration

This chapter describes the basics of administering Sun Java System Application Server: where to find general information about the server, how to access the server's user interfaces, and how to access online help.

This chapter includes the following sections:

- [About Sun Java System Application Server](#)
- [Using the Administration Interface](#)
- [Using the Command-line Interface](#)
- [Accessing the Administration Server](#)
- [Accessing Sun Java System Application Server Instances](#)
- [Using Sun ONE Studio](#)
- [About Configuration Files](#)
- [Using the License Commands](#)

## About Sun Java System Application Server

The Sun Java System Application Server provides a robust J2EE platform for the development, deployment, and management of e-commerce application services to a broad range of servers, clients, and devices. Key features include transaction management, performance, scalability, security, and enterprise application integration.

Sun Java System Application Server supports services from web publishing to enterprise-scale transaction processing, while enabling developers to build applications based on JavaServer Pages (JSP™), Java Servlet and Enterprise JavaBeans™ (EJB™) technology.

It contains the following basic tools for administrators:

- Multiple administrative domains that allow different administrators to create and manage their own set of Sun Java System Application Server instances.
- An Administration Server that provides administration facilities (one Administration Server per domain)
- A graphical user interface for server administration (the Administration interface)
- A command-line interface you can use to perform the same tasks as the Administration interface

Use these tools to perform all administrative functions, including:

- Administering domains
- Managing server instances
- Deploying applications
- Monitoring the server
- Using log files
- Administering resources
- Administering a Message Queue server
- Using transaction services
- Using Corba/IIOP clients
- Configuring the web server plug-in
- Configuring J2EE containers
- Administering HTTP server features

For more information about the Sun Java System Application Server architecture and features, and for initial steps to take with your Sun Java System Application Server, see the *Sun Java System Application Server Getting Started Guide*.

# Using the Administration Interface

Use the Administration interface to configure all aspects of your server. This section contains the following topics:

- [Accessing the Administration Interface](#)
- [Using Tabs](#)
- [Using Buttons](#)
- [Accessing Online Help](#)
- [Exiting the Administration Interface](#)

---

**NOTE** Some aspects of server configuration and corresponding Administrative 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 when they do occur.

---

## Accessing the Administration Interface

The Administration interface for Sun Java System Application Server runs on an HTTP server called the Administration Server. The Administration Server is installed when you install Sun Java System Application Server.

The Administration Server must be running in order for you to use the Administration interface. For information on starting the Administration Server, see [“Starting the Administration Server” on page 52](#).

When you installed the Sun Java System Application Server or created a domain, you chose a port number for the Administration Server, or used the default port of 4848. To access the Administration interface, in a web browser type:

`http://hostname.domain:port/`

For example:

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

If you are accessing the Administration Server from the same machine you installed Sun Java System Application Server on, you can use:

`http://localhost:4848`

You are prompted for a user name and password, which the installing user set during installation or when you created the domain and Administration Server.

You can access the Administration interface from a remote location as long as you have access to a browser. You can access it from any machine that can reach the server over the network.

On Windows, you can access the Administration interface from the Start menu by choosing Programs, then Sun Microsystems, then Sun Java System Application Server 7 2004Q2, then Start Admin Console.

The left pane is a tree view of all items you can configure in the Sun Java System Application Server. To use the Administration interface, click an item in the left pane. The right pane displays the page associated with that item.

If an item in the left pane has an open/close symbol next to it, you can expand that item to subitems by clicking on the open/close symbol.

When the tree item is *not* expanded, the symbol looks like this:

**Figure 1-1** Closed Symbol



When the tree item is expanded, the symbol looks like this:

**Figure 1-2** Open Symbol



## Using Tabs

Some portions of the Administration interface contain tabs which you use to navigate to other pages. These tabs appear in a separate pane at the top of the right pane.

To use the tabs, click the tab name. Some tabs take you directly to a page, while others have a list of pages available from the tab which appear below the tab names. Click a page name to go to that page.

## Using Buttons

The following standard buttons are available in the Administration interface.

**Table 1-1** Standard Administration interface Buttons

Button	Action Performed
Cancel	Cancels without saving your changes and returns to previous page.
Delete	Deletes an item. Often you click Select next to an item to indicate what to delete when you click Delete.
New	Displays a page for creating a new item. For example, clicking New on the Sun Java System Application Server Instances page displays the Create New Instance page.
OK	Saves your entries. If you've made configuration changes to a Sun Java System Application Server instance, you must apply your changes in order for them to take effect.  For more information, see <a href="#">“Applying Changes to an Application Server Instance” on page 83</a> .
Reset	Resets the values on the page to the default values.
Save	Saves your entries. If you've made configuration changes to a Sun Java System Application Server instance, you must apply your changes in order for them to take effect.  For more information, see <a href="#">“Applying Changes to an Application Server Instance” on page 83</a> .

Other buttons are available depending upon the needs of a particular screen.

## Accessing Online Help

You can access help for any page in the Administration interface by clicking the Help button in the banner at the top of the Administration interface. The online help describes the use of the page you are accessing and gives information about what to enter in the fields on the page.

You can navigate to help for other pages using the table of contents in the left pane of the help window. For help on using help, see the first topic, Using Help, in the online help table of contents.

## Exiting the Administration Interface

There is no explicit exit or logout button in the Administration interface. To exit, close the browser you are using to access the Administration interface. Also, close other instances of the same browser that may be running on your machine.

## Using the Command-line Interface

Sun Java System Application Server contains a command-line interface. You can use the `asadmin` utility and the commands associated with it to perform the same set of tasks as you can perform in the Administration interface. For example, you can start and stop Sun Java System Application Server instances, configure the server, and deploy applications.

You can use these commands either from a command prompt in the shell, or you can call them from other scripts and programs. You can use these commands to automate repetitive administration tasks.

For more information about using the command-line interface, and for a list of commands, see [Appendix A, “Using the Command Line Interface.”](#)

## Accessing the Administration Server

The Administration Server is a special instance of the Sun Java System Application Server that serves the Administration interface and provides administrative facilities for the Administration interface and the command-line interface. It manages the configuration, deployment, and monitoring facilities for these interfaces and tools, and so must be running for them to work.

To configure Administration Server properties, access the Administration interface. Click Admin Server in the left pane to display configuration settings for the Administration Server.

For more information on the Administration Server, see [Chapter 2, “Setting Administration Server Preferences.”](#)



# Accessing Sun Java System Application Server Instances

You can have multiple application server instances. Each application server instance has its own configuration, resources, and application deployment areas that are independent of other application server instances. If you change the configuration for one application server instance, you need not change the configurations of other application server instances. The Administration interface contains an item for every application server instance you create. One application server instance is created at installation time. You can create others if you like.

For more information on application server instances, see [Chapter 4, “Using Sun Java System Application Server Instances.”](#)

## Using Sun ONE Studio

To deploy applications, in addition to using the Administration interface and the command-line interface, you can also use Sun ONE Studio 5 Enterprise Edition Update 1.

For more information about Sun ONE Studio, see the Sun ONE Studio 5, Enterprise Edition Update 1 documentation at <http://docs.sun.com>.

## About Configuration Files

When you make a change to the server settings through the Administration interface or command-line interface, the Administration Server changes the underlying configuration files. These files contain setting for the Administration Server and for individual application server instances.

For more information on the files and what settings they contain, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

# Using the License Commands

When you purchase Sun Java System Application Server, a license of the appropriate type is installed automatically during installation. For more information on the types of licenses and their restrictions, see the *Sun Java System Application Server Installation Guide*.

Sun Java System Application Server contains command-line utilities for managing your licenses after installation.

To upgrade your license after installation, you can use the `asadmin` utility's `install-license` command for Windows machines. This command must be run locally on the machine on which you have installed the Sun Java System Application Server. You must stop any running application server before installing a license.

It has the following syntax:

```
asadmin install-license
```

The following syntax is used for license installation:

```
pkgadd -d full_path SUNWaslco
```

For example,

```
pkgadd -d /install_dir/pkg SUNWaslco
```

To get information on your license, use the `display-license` command, which has the following syntax:

```
asadmin display-license [--user admin_user] [--password admin_password]
[--passwordfile password_file] [--host localhost] [--port admin_port]
[--local=true/false]
```

This command can be run locally or remotely, depending on the value of the `local` option. For example, to run the command from the local machine, taking the defaults for the host and the port number, the syntax is:

```
asadmin display-license --local
```

The output describes the type of license currently installed (for example, evaluation) the expiration date, if it has one, the number of instances per Administration Sever your license allows, and whether remote administration is allowed or not.

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

# Setting Administration Server Preferences

This chapter includes the following topics:

- [About the Administration Server](#)
- [Starting the Administration Server](#)
- [Shutting Down the Administration Server](#)
- [Accessing the Administration Server Settings](#)
- [Viewing Administration Server Control Settings](#)
- [Applying Changes to the Administration Server](#)
- [Editing HTTP Listener Settings for the Administration Server](#)
- [Setting SNMP, Logging, and Security Preferences](#)

## About the Administration Server

The Administration Server is a special instance of the Sun Java System Application Server that provides administrative facilities for the Administration interface and the command-line interface. It manages the configuration, deployment, and monitoring facilities for these interfaces. It serves the Administration interface pages. It must be running in order for you to use the Administration interface and to run most commands in the command-line interface.

An Administration Server is installed when you install the Sun Java System Application Server or create a new domain. You can only have one Administration Server per domain, though you may create multiple Sun Java System Application Server instances that are managed by an Administration Server. You can access configuration settings, deployed applications, and other server features for each Sun Java System Application Server instance by using the Administration Server.

For more information on administrative domains, see [Chapter 3, “Configuring Administrative Domains.”](#)

When you installed the Sun Java System Application Server you chose a port number for the Administration Server, or used the default port of 4848.

To access the Administration interface, in a web browser type:

<http://hostname.domain:port/>

Note that *domain* here is not your Sun Java System Application Server administrative domain, but your domain name.

For example:

<http://austen.sun.com:4848/>

If you are accessing the Administration Server from the same machine you installed Sun Java System Application Server on, you can use:

<http://localhost:4848>

You are prompted for a user name and password.

## Starting the Administration Server

To start or restart the Administration Server, use one of the methods described in the following topics:

- [Starting Using the startserv Script](#)
- [Starting Using the Command-Line Interface](#)
- [Starting Using the Services Window \(Windows\)](#)
- [Starting Using the Start Menu \(Windows\)](#)

## Starting Using the startserv Script

To start the Administration Server using the start script, log in as root if the server runs on ports with numbers lower than 1024 (UNIX); otherwise, log in as root or with the server's user account. At the command-line prompt, go to the directory:

```
install_dir/domains/domain_dir/admin-server/bin
```

where *install\_dir* is the directory where you installed the server, and *domain\_dir* is the administrative domain directory.

For UNIX, type:

```
./startserv
```

You can use the optional parameter `-i` at the end of the line. The server normally runs as a background process. The `-i` option prevents the server from putting itself in the background. This option is useful if you are running the server using `/etc/inittab`.

For Windows, run the `startserv.bat` file.

---

<b>NOTE</b>	If the server is already running, the <code>startserv</code> command will fail. You must stop the server first, then use the <code>startserv</code> command. Also, if the server startup fails, you should kill the process before trying to restart it.
-------------	--

---

## Starting Using the Command-Line Interface

The command-line interface's `asadmin` utility has a command `start-domain` that you can use to start your Sun Java System Application Server and all associated Sun Java System Application Server instances. You can only run this command locally, that is, from the machine where your Sun Java System Application Server is installed. This command requires no arguments.

You can also use the command `start-domain` to start all instances within an administrative domain. It uses the syntax:

```
asadmin start-domain [--domain domain-name]
```

For more information on using the command-line interface, see the online help for the command-line interface and [Appendix A, "Using the Command Line Interface."](#)

## Starting Using the Services Window (Windows)

To start the server by using the Services Control Panel in Windows, follow these steps:

1. In the Control Panel click Administrative Tools.
2. Click Services.
3. Scroll through the list of services and double-click the Sun App Server Admin Server service.

Select the Administration Server for the given domain. The Name column in the Services window displays the name of an Admin Server as Sun App Server Admin Server (*your\_domain\_name*:admin-server).

4. Click Start.
5. Click OK.

The Admin Server service starts automatically when you start your computer.

## Starting Using the Start Menu (Windows)

To start the server by using the Windows Start menu, follow these steps:

1. From the Start menu, choose Programs.
2. Choose Sun Microsystems.
3. Choose Sun Java System Application Server 7 2004Q2
4. Click Start Application Server.

## Shutting Down the Administration Server

Once the Administration Server is started it runs constantly, listening for and accepting requests. You might want to stop and restart your server if, for instance, you change the Administration Server logging preferences or the port that the Administration Server's HTTP Listener listens on.

When you stop the Administration Server, it stops accepting new connections. Then it waits for all outstanding connections to complete. While the Administration Server is stopped, you cannot use the Administration interface or command-line interface.

You can stop the server using one the methods described in the following topics:

- [Shutting Down Using the Administration Interface](#)
- [Shutting Down Using the stopserv Script](#)
- [Shutting Down Using the Command-Line Interface](#)
- [Shutting Down Using the Services Window \(Windows\)](#)

After you shut down the server, it may take a few seconds for the server to complete its shut-down process.

## Shutting Down Using the Administration Interface

To shut down the Administration Server using the Administration interface, follow these steps:

1. In the left pane, click Admin Server.
2. Click the Control tab.
3. Click Stop.

The Administration Server shuts down immediately when you click this link. There is no additional screen.

## Shutting Down Using the stopserv Script

To stop the Administration Server manually, at the command prompt go to the directory:

```
install_dir/domains/domain_dir/admin-server/bin
```

where *install\_dir* is the directory where you installed the server and *domain\_dir* is the domain directory.

For UNIX, type:

```
./stopserv
```

You must run this command as the user that is running the server.

If you used the `*/etc*/inittab` file to restart the server you must remove the line starting the server from `*/etc*/inittab` and type `kill -1` before you try to stop the server. Otherwise, the server restarts automatically after it is stopped.

For Windows, run the `stopserv.bat` file.

## Shutting Down Using the Command-Line Interface

You can stop the Administration Server using the command-line interface `asadmin` utility's `shutdown` command. This command requires no arguments and can be run either locally or remotely.

You can also stop the Administration Server and all the associated Sun Java System Application Server instances using the command-line interface `asadmin` utility's `stop-appserv` command. You can only run this command locally, that is, from the machine where your Sun Java System Application Server is installed. This command requires no arguments.

You can also shut down the Administration Server by shutting down the domain using the `stop-domain` command. This command shuts down all instances in the domain, including the Administration Server, by default. You can also configure it to shut down all instances in the domain *except* the Administration Server. This command has the following syntax:

```
asadmin stop-domain [--user admin_user] [--password admin_password] [--host
admin_host] [--port admin_port] [--local=true/false] [--domain domain_name]
[--adminserv=true/false] [--passwordfile file_name] [--secure | -s]
```

If you use the `local` option, the command runs locally. If you use the `--adminserv=false` option, the command will not stop the Administration Server. However, `--adminserv` is set to `true` by default, so the Administration Server is stopped by default.

For more information on using the command-line interface, see the online help for the command-line interface and [Appendix A, “Using the Command Line Interface.”](#)



## Shutting Down Using the Services Window (Windows)

To shut down the Administration Server using the Services window, follow these steps:

1. In the Control Panel click Administrative Tools.
2. Click Services.
3. Scroll through the list of services and double-click the Sun App Server 7 Admin Server service.
4. Click Stop.
5. Click OK.

## Accessing the Administration Server Settings

To access the Administration Server settings, in the left pane of the Administration interface click Admin Server. The settings for the Administration Server appear in the right pane, organized by a set of tabs.

Click a tab to access the settings for a particular functional area. Sometimes clicking a tab takes you directly to a page, and sometimes clicking a tab gives you a list of pages to choose from.

This chapter covers the Control and HTTP Listener tabs. For information on Monitoring and SNMP settings, see [Chapter 6, “Monitoring Sun Java System Application Server.”](#) For information on logging, see [Chapter 5, “Using Logging.”](#)

## Viewing Administration Server Control Settings

The Administration Server control settings show the instance name (Admin Server), what port your Administration Server runs on, what directory contains the configuration files, and what version of the Sun Java System Application Server software you are running.

To view these settings:

1. In the left pane, click Admin Server.
2. Click the Control tab.

# Applying Changes to the Administration Server

When you change the Administration Server's configuration information using the Administration interface or the command line interface you may need to apply your changes in order for your changes to take effect. This is also called reconfiguring the server. When you apply your changes, all changes made to the configuration since the last time you applied changes take effect.

If you've made changes to the Administration Server's configuration that require you to apply changes, a yellow icon appears next to the Admin Server in the left pane's tree view.

**Figure 2-1** Warning Icon



To apply configuration changes:

1. In the left pane, click Admin Server.
2. Click the Control tab.
3. Click Apply Changes.

When the changes are applied, the screen displays a message.

## Editing HTTP Listener Settings for the Administration Server

Before the server can process a request, it must accept the request via an HTTP listener.

HTTP listener for the Administration Server, `http-listener-1`, is created automatically at the time of installation. This HTTP listener uses the IP address `0.0.0.0` and the port number you specified as your Administration Server port number during installation. The default is 4848. You cannot delete the default HTTP listener.

For an administrative server instance (in a domain), only the HTTP listener has the `http-listener-1` ID. In other words, if you create an administration server instance, only one HTTP listener can act in either HTTP or HTTPS protocol at any point in time. (It also means that you cannot have both HTTP and HTTPS connections to an administration server simultaneously.) The HTTP listener is where you activate and configure SSL/TLS security settings for your Administration Server.

In addition, you specify the number of acceptor threads (sometimes called accept threads) in the HTTP listener. Accept threads are threads that wait for connections. The threads accept connections and put them in a queue where they are then picked up by worker threads. Ideally, you want to have enough accept threads so that there is always one available when a new request comes in, but few enough so that they do not provide too much of a burden on the system. The default is 1. A good rule is to have one accept thread per CPU on your system. You can adjust this value if you find performance suffering. For more information on performance, see the *Sun Java System Application Server Performance Tuning Guide*.

To edit your Administration Server's HTTP listener settings:

1. In the left pane, click Admin Server.
2. Click the HTTP Listeners tab.
3. Make the desired changes and click OK.

For more information on HTTP listeners, see the online help.

## Setting SNMP, Logging, and Security Preferences

For information on SNMP settings, see [Chapter 6, “Monitoring Sun Java System Application Server.”](#) For information on logging, see [Chapter 5, “Using Logging.”](#) For information on security settings, see the *Sun Java System Application Server Administrator's Guide to Security*.



# Configuring Administrative Domains

This chapter explains how to set up and administer administrative domains with your Sun Java System Application Server.

This chapter includes the following topics:

- [About Administrative Domains](#)
- [Configuring Domains](#)
- [Recreating the Domain Registry](#)

## About Administrative Domains

Administrative domains provide a basic security structure whereby different administrators can administer specific groups (domains) of application server instances on a machine. By dividing the application server instances up like this, it is possible to share a single machine between disparate organizations, each with their own administrator.

Within the Sun Java System Application Server every application server instance is a member of one domain. It is not required that there be any more than one domain, however, multiple domains are supported as a useful feature if so desired.

Administrative security is established for local commands using the underlying operating system's security mechanisms (that is, via file permissions). Remote command security is established using a username/password pair to communicate with a specific admin server. Administrative domains do not utilize any other security constructs.

This section describes the following topics:

- [Implementing Administrative Domains](#)
- [Directory Structure](#)
- [Process/Port Structure](#)

## Implementing Administrative Domains

A domain is implemented using files, operating system processes and ports. Each domain has a unique name.

### Directory Structure

There are files (configuration, executables, and so on) that are shared by all domains within an installation. What is important for this discussion are those files which are specific to a domain.

The files specific to a domain all share a common root directory known as the domain directory, and whose name is the name of the domain. Under the domain directory is one directory per instance, each named after the instance, and under each of those instance directories are instance-specific files.

A domain directory can be constructed anywhere in the file system (in accordance with security permissions and other operating system level constraints). Unless a user chooses otherwise, domain directories are constructed under a default directory (known as the domains directory). However a user can choose to create a domain directory anywhere.

### Process/Port Structure

When a domain is running it consumes operating system processes and ports. Specifically, for each instance running within a domain (including the domain's admin server) there is a process and a port.

## Configuring Domains

Domains can be created, deleted, listed, started and stopped using commands specifically designed for that purpose.

Creation, deletion and starting of domains can only be done locally, whereas listing and stopping can be done both locally and remotely.

The deletion, starting and stopping commands all take a domain name. This name is optional if there is only one domain. The command will give an error if no domain was given but there are multiple domains configured.

This section contains the following topics:

- [Creating Domains](#)
- [Deleting Domains](#)
- [Listing Domains](#)
- [Starting Domains](#)
- [Stopping Domains](#)

## Creating Domains

Domains are created using the `create-domain` command. This command is local only.

Synopsis:

```
asadmin create-domain [--path domain_path] [--sysuser sys_user] [--passwordfile
file_name] --adminport port_number --adminuser admin_user --adminpassword password
domain_name
```

### Example: creating a domain in the default location

```
$ asadmin create-domain --adminport 123 --adminuser MyAdmin
--adminpasswd MyPassword MyDomain
```

This example creates a domain called `MyDomain` in the default location (that is, the domains directory). The administration server will listen on port 123, the admin user name will be `MyAdmin` and the password will be `MyPassword`. The domain directory and files underneath it will be owned by the operating system user who executed this command. In addition, the operating system processes will run as the user who executed this command.

If there's already a domain called `MyDomain` then an error message is returned.

(Note that instead of using the password on the command line, which could be a security issue, you can put the password in a file and pass it through in using the `--passwordfile` option).

### Example: creating a domain somewhere other than the default location

```
$ asadmin create-domain --path $HOME --adminport 123 --adminuser
MyAdmin --adminpasswd MyPassword MyDomain
```

This example is similar to the first, except that the domain directory will now be located underneath the user's `$HOME` directory rather than under the default domains directory.

### Example: creating a domain for another user (UNIX only)

```
# asadmin create-domain --user AnotherUser --adminport 123 --adminuser
MyAdmin --adminpasswd MyPassword MyDomain
```

This example is similar to the first, except that the domain and its files will be owned by the user, `AnotherUser`, as will the operating system processes. Using the `--user` option provides the ability for one user to construct domains that other users can subsequently administer. This option requires that the user running the `create-domain` command be root.

## User Permissions on UNIX Platforms

In order for a non-root user to create and delete administrative domains, you must add the user ID to a UNIX group that has write permissions to the domain configuration files:

1. Create a UNIX group that will be the group which is applied to the installation-wide domain configuration files. For example, a UNIX group named `asadmin`.
2. Set the installation-wide domain configuration files located under `/etc/appserver` to be owned by the newly created UNIX group.

The files are named `domains.bin` and `domains.lck`. For example, after changing the group assigned to these files:

```
-rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.bin
-rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
```

3. Enable write access to these files by the newly created UNIX group. In this example, the resulting permissions would look like the following:

```
-rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.bin
-rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
```

4. Add the user ID to the UNIX group.



Alternatively, if you do not want to provide non-root users with write access to the installation-wide configuration files, you can create an administrative domain on behalf of a user. During creation of a new administrative domain, specify the `--sysuser` and `--path` options to identify the UNIX user ID that will own the domain's directories and files and the location under which the administrative domain will be created. For an example, see [“Example: creating a domain for another user \(UNIX only\)” on page 64](#).

Once an administrative domain is created under a user ID, the user may create new application server instances and perform a wide variety of administrative operations on the application server instances. The user ID does not need to belong to the UNIX group that has write privileges for the administrative domain configuration file. Membership in the UNIX group is required only to create and delete administrative domains.

## Deleting Domains

Domains are deleted using the `delete-domain` command. Only the operating system user (or root) who can administer the domain can execute this command successfully. This command is local only.

Synopsis:

```
asadmin delete-domain [domain_name]
```

### Example: deleting a domain

```
$ asadmin delete-domain MyDomain
```

This example deletes the domain called `MyDomain` on the local machine.

## Listing Domains

The domains created on a machine can be found using the `list-domains` command. This command can operate both locally and remotely.

Synopsis:

```
asadmin list-domains [--host host] [--port port] [--password password] [--user user]
```

### Example: listing the domains on a local machine

```
$ asadmin list-domains

domain1      [/opt/ias/build/domains/domain1]
```

### Example: listing the domains on the local machine using the remote option:

```
$ asadmin list-domains --user admin --password password --host
localhost --port 4848

domain1      [/opt/ias/build/domains/domain1]
```

## Starting Domains

Domains can be started using the `start-domain` command. This starts up the domain's admin server and all other instances in the domain. This command can only be run locally.

### Synopsis:

```
asadmin start-domain [--domain domain_name]
```

### Example: starting the only domain on a machine:

```
$ start-domain

Instance domain1:admin-server started
Instance domain1:server1 started
Domain domain1 Started.
```

## Stopping Domains

Domains can be stopped using the `stop-domain` command. The user can choose to stop every instance within a domain, or all the instances except the admin server, thus leaving the domain able to be remotely administered.

This command can be run both locally and remotely.

**Synopsis:**

```
asadmin stop-domain [--user admin_user] [--password admin_password] [--host
host_name] [--port port_name] [--local=false] [--domain domain_name]
[--adminserv=true] [--passwordfile file_name] [--secure | -s]
```

**Example: Stopping all instances in a domain except for the admin server instance.**

```
$ asadmin asadmin stop-domain --user admin --password password --host
localhost --port 4848 --adminserv=false --domain domain1
```

```
DomainStoppedRemotely
```

## Recreating the Domain Registry

For implementation purposes the details of each domain (its name, location, ports used, and so on) are recorded in a file known as the domain registry.

Under normal operating conditions you should not have to do anything with the domain registry directly, since any modification or use of the domain registry is entirely encapsulated by the commands used to administer the system. However, because the domain registry is a file, it can become corrupted (for example, when someone inadvertently deletes the registry, and so on), and in those cases you may have to recreate the file.

---

**NOTE** You can access the domain registry through the command line interface by using the `asadmin` command.

---

If the registry becomes corrupted, perform the following procedure to recreate the registry:

1. Get a list of all the domains, and the directory (default or non-default) that they are located in.
2. Rename each directory (for example, append each directory name with the suffix “.bak”)
3. Create each domain again in its original location, using default values for ports, passwords, and so on.
4. Delete each new domain directory and replace it with the original directory.

5. For each domain execute the `reconfig` command. This will cause the domain registry to be updated with the values from the old domain.

# Managing an Individual Server Instance

Chapter 4, “Using Sun Java System Application Server Instances”

Chapter 5, “Using Logging”

Chapter 6, “Monitoring Sun Java System Application Server”

Chapter 7, “Configuring J2EE Containers”

Chapter 8, “Using Transaction Services”

Chapter 9, “Configuring Naming and Resources”

Chapter 10, “Using the JMS Service”

Chapter 11, “Configuring the Server For Corba/IIOP Clients”

Chapter 12, “Configuring the Web Server Plug-in”

Chapter 13, “Deploying Applications”



# Using Sun Java System Application Server Instances

This chapter describes how to create, delete, configure, start, and stop Sun Java System Application Server instances.

This chapter includes the following topics:

- [About Sun Java System Application Server Instances](#)
- [Starting and Stopping an Application Server Instance](#)
- [Starting the Application Server Instance in Debug Mode](#)
- [Setting the Termination Timeout](#)
- [Restarting an Application Server Instance Automatically \(UNIX\)](#)
- [Restarting an Application Server Instance Manually \(UNIX\)](#)
- [About the Watchdog](#)
- [Adding an Application Server Instance](#)
- [Deleting an Application Server Instance](#)
- [Applying Changes to an Application Server Instance](#)
- [Viewing Application Server Instance Status](#)
- [Configuring JVM Settings](#)
- [Configuring Logging Setting and Monitoring Settings](#)
- [Changing Application Server Instance Advanced Settings](#)

# About Sun Java System Application Server Instances

Sun Java System Application Server creates one application server instance, called `server1` at the time of installation. You can delete the `server1` instance and create a new instance with a different name if you prefer.

Each Sun Java System Application Server instance has its own J2EE configuration, J2EE resources, application deployment areas, and server configuration settings. Changes to one application server instance have no effect on other application server instances. You can have many application server instances within one administrative domain. Within a domain, all the server instances have the same Administration Server. For more information on domains, see [Chapter 3, “Configuring Administrative Domains.”](#)

For many users, one application server instance meets their needs. However, depending upon your environment, you might want to create one or more additional application server instances. For example, in a development environment you can use different application server instances to test different Sun Java System Application Server configurations, or to compare and test different application deployments. Because you can easily add or delete an application server instance, you can use them to create temporary “sandbox” areas to experiment with while developing.

In addition, for each application server instance you can also create virtual servers. Within a single installed application server instance you can offer companies or individuals domain names, IP Addresses, and some administration capabilities. For the users, it is almost as if they have their own web server, without the hardware and basic server maintenance. These virtual servers do not span application server instances. For more information about virtual servers, see [Chapter 15, “Using Virtual Servers.”](#)

In operational deployments, for many purposes you can use virtual servers instead of multiple application server instances. However, if virtual servers do not meet your needs, you can also use multiple application server instances.



# Starting and Stopping an Application Server Instance

A Sun Java System Application Server instance is not started automatically. Once you start an instance, the instance runs until you stop it. When you stop an application server instance, it stops accepting new connections, then waits for all outstanding connections to complete. If your machine crashes or is taken offline, the server quits and any requests it was servicing may be lost.

You can start and stop the application server instance using one of several methods, covered in the following topics:

- [Using the Start and Stop Buttons in the Administration Interface](#)
- [Using the start-instance and stop-instance Commands](#)
- [Using the Windows Services \(Windows\)](#)
- [Using the startserv and stopserv Scripts](#)

---

<b>NOTE</b>	If you have a security module installed with your server, you will be required to enter the appropriate passwords before starting or stopping the server.
-------------	---

---

If you have a server certificate installed, the Sun Java System Application Server prompts the administrator for the key database password before starting up. If you want to be able to restart an unattended Sun Java System Application Server, you need to save the password in a `password.conf` file. Only do this if your system is adequately protected so that this file and the key databases are not compromised. For more information on creating and using `password.conf`, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

---

<b>NOTE</b>	On UNIX, some Sun Java System Application Server installations may require access to more memory and/or file descriptors than your operating system allows by default. If you are unable to start the server, check the resource limits imposed by your operating system using the <code>ulimit</code> command. Your operating system's <code>ulimit</code> man page should provide more information.
-------------	---

---

## Using the Start and Stop Buttons in the Administration Interface

To start and stop the server using the Administration interface:

1. In the left pane, under App Server Instances, click the name of the instance to start or stop.
2. In the right pane, click Start or Stop; or in the General Tab, click Start or Stop.

You see a message when the application server instance is started or stopped successfully.

## Using the start-instance and stop-instance Commands

Using the command-line interface utility `asadmin`, you can start and stop your application server instances either from the command prompt or from a script. Use the commands `start-instance` and `stop-instance`.

These commands have the following syntax:

```
start-instance [--user admin_user] [--password admin_password] [--host admin_host]  
[--port admin_port] [--local=true/false] [--domain domain_name] [--debug=true/false]  
[--passwordfile file_name] [--secure | -s] instance_name
```

```
stop-instance [--user admin_user] [--password admin_password] [--host admin_host]  
[--port admin_port] [--local=true/false] [--domain domain_name] [--passwordfile  
file_name] [--secure | -s] instance_name
```

These commands have a `local` option which you can use to start or stop the server without going through the Administration Server. If you use the `local` option, you do not need to specify the `host`, `port`, `user`, and `password` (or `passwordfile`) options.

For information on the syntax of these commands, use the `asadmin help`. For information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Using the Windows Services (Windows)

You can start the server by using the Services Control Panel in Windows.

Follow these steps:

1. In the Control Panel click Administrative Tools.
2. Click Services.
3. Scroll through the list of services and double-click the service for your server.  
It is called Sun App Server 7 (*domain\_name:instance\_name*). For example, Sun App Server 7 (domain1:server1).
4. Click Start or Stop.
5. Click OK.

## Using the startserv and stopserv Scripts

To use the `startserv` and `stopserv` scripts, at the command-line prompt go to the directory:

*instance\_dir*/bin

where *install\_dir* is the directory where you installed the server, *domain\_dir* is the domain directory, and *instance\_dir* is the name of the instance you want to start.

For UNIX, type:

```
./startserv
```

Log in as root if the server runs on ports with numbers lower than 1024; otherwise, log in as root or with the server's user account.

You can use the optional parameter `-i` at the end of the line. The `-i` option runs the server in `inittab` mode, so that if the server process is ever killed or crashed, `inittab` will restart the server for you. This option also prevents the server from putting itself in a background process.

---

<b>NOTE</b>	If the server is already running, the <code>startserv</code> command will fail. You must stop the server first, then use the <code>startserv</code> command. Also, if the server startup fails, you should kill the process before trying to restart it.
-------------	--

---

For Windows, run the `startserv.bat` file.

To stop the server manually, at the command-line prompt go to the directory:

*instance\_dir*/bin

where *install\_dir* is the directory where you installed the server and *instance\_dir* is the name of the instance you want to start.

For UNIX, type:

```
./stopserv
```

If you used the `/etc/inittab` file to restart the server you must remove the line starting the server from `/etc/inittab` and type `kill -1 1` before you try to stop the server. Otherwise, the server restarts automatically after it is stopped.

For Windows, run the `stopserv.bat` file.

## Starting the Application Server Instance in Debug Mode

You can run the application server instance in debug mode if developers want to debug their J2EE applications.

To start the server in debug mode:

1. Access the Administration interface and click the name of the application server instance you want to start in debug mode.
2. Click the General tab.
3. Click the checkbox next to Run in Debug Mode.
4. Restart the application server instance.

Debug mode changes the JVM settings. Debug Enabled is set to true, and the Debug Options change. For more information on the JVM Debug options, see the Java Platform Debugger Options documentation at <http://java.sun.com/products/jpda/doc/conninv.html>.

To start the application server instance in debug mode from the command-line interface, use the `asadmin` utility's `start-instance` command with the `debug` option set to true. For more information on the command syntax, see the online help for the command-line interface.

## Setting the Termination Timeout

When you stop an application server instance, it stops accepting new connections. Then it waits for all outstanding connections to complete. The time the server waits before timing out is configurable in the `init.conf` file, which can be found in `instance_dir/config/`. By default it is set to 30 seconds. To change the value, add the following line to `init.conf`:

```
TerminateTimeout seconds
```

where *seconds* represents the number of seconds the server will wait before timing out.

The advantages to configuring this value is that the server will wait longer for connections to complete. However, because servers often have connections open from nonresponsive clients, increasing the termination timeout may increase the time it takes for the server to shut down.

## Restarting an Application Server Instance Automatically (UNIX)

You can restart an application server instance using one of the following methods:

- Automatically restart it from the `/etc/inittab` file.  
Note that if you are using a version of UNIX not derived from System V, you will not be able to use the `/etc/inittab` file.
- Automatically restart it with daemons in `/etc/rc2.d` when the machine reboots.
- Restart it manually. See [“Starting and Stopping an Application Server Instance” on page 73](#) and [“Deleting an Application Server Instance” on page 82](#).

This section contains the following topics:

- [About Restarting Automatically](#)
- [Restarting Automatically with `/etc/inittab` \(UNIX\)](#)
- [Restarting Automatically with the System RC Scripts \(UNIX\)](#)

## About Restarting Automatically

Because the installation scripts cannot edit the `/etc/rc.local` or `/etc/inittab` files, you must edit those files with a text editor. If you do not know how to edit these files, consult your system administrator or system documentation.

Normally, you cannot start an SSL-enabled server with either of these files because the server requires a password before starting. Although you can start an SSL-enabled server automatically if you keep the password in plain text in a file, this is *not* recommended.

---

**CAUTION** Leaving the SSL-enabled server's password in plain text in the server's `startserv` script is a large security risk. Anyone who can access the file has access to the SSL-enabled server's password. Consider the security risks before keeping the SSL-enabled server's password in plain text.

---

The server's `startserv` script, key pair file, and the key password should be owned by root (or, if a non-root user installed the server, that user account), with only the owner having read and write access to them.

## Restarting Automatically with `/etc/inittab` (UNIX)

To restart the server using `inittab`, put the following text on one line in the `/etc/inittab` file:

```
http:2:respawn:install_dir/path_to_domain_dir/instance_dir/bin/startserv -start -i
```

where *install\_dir* is the directory where you installed the server, *path\_to\_domain\_dir* is the path to the domain and *instance\_dir* is the server's directory.

The `-i` option prevents the server from putting itself in a background process.

You must remove this line before you stop the server, otherwise the server automatically restarts.

## Restarting Automatically with the System RC Scripts (UNIX)

If you use `/etc/rc.local`, or your system's equivalent, place the following line in `/etc/rc.local`:

```
install_dir/path_to_domain_dir/instance_dir/bin/startserv
```

Replace *install\_dir* with the directory where you installed the server, *path\_to\_domain\_dir* with the path to the domain, and *instance\_dir* with the name of the application server instance.

## Restarting an Application Server Instance Manually (UNIX)

On UNIX, you have the option of restarting the server instance manually. Unlike stopping the server instance, then starting it, a restart does not stop the watchdog program. For information about the watchdog, see [“About the Watchdog” on page 81](#).

---

**NOTE** If you have made manual changes to your configuration files by editing them, you must apply changes before you restart the server, either by using the Apply Changes button in the Administration interface, or by using the `asadmin reconfig` command with the `keepmanualchanges` option set to `true`. For more information on applying changes, see [“Applying Changes to an Application Server Instance” on page 83](#).

---

There are three ways to restart the server instance, covered in the following topics:

- [Restarting the Server Instance Using the Restart Button \(UNIX\)](#)
- [Restarting the Server Instance Using the restart-instance Command \(UNIX\)](#)
- [Restarting the Server Instance Using the restartserv Script \(UNIX\)](#)

## Restarting the Server Instance Using the Restart Button (UNIX)

To restart the server instance using the Administration interface:

1. In the left pane, under App Server Instances, click the name of the instance to restart.
2. In the right pane, click Restart.

You see a message when the application server instance is restarted successfully.

## Restarting the Server Instance Using the restart-instance Command (UNIX)

Using the command-line interface utility `asadmin`, you can start and stop your application server instances either from the command-line or from a script. Use the commands `restart-instance`. This command has the following syntax:

```
restart-instance [--user admin_user] [--password admin_password] [--host  
admin_host] [--port admin_port] [--local=true/false] [--domain domain_name]  
[--passwordfile file_name] [--secure | -s] instance_name
```

This command has a `local` option which you can use to restart the server instance without going through the Administration Server.

For information on the syntax of these commands, use the `asadmin help`. For information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Restarting the Server Instance Using the restartserv Script (UNIX)

To use the `restartserv` script, at the command-line prompt go to the directory:

`install_dir/bin`

where *install\_dir* is the directory where you installed the server, *domain\_dir* is the domain directory, and *instance\_dir* is the name of the instance you want to start.



Type:

```
./restartserv
```

Log in as root if the server runs on ports with numbers lower than 1024; otherwise, log in as root or with the server's user account.

## About the Watchdog

The watchdog (`appserv-wdog` on UNIX, `appservd-wdog.exe` on Windows) is a program that is shipped with your Sun Java System Application Server. It performs the following tasks:

- Starts the server
- Stops the server
- If SSL/TLS is enabled, prompts the administrator for the trust database password when the server starts
- Restarts the server if it goes down

The watchdog runs in the background, without user intervention. You should never have to configure or otherwise change it. One watchdog runs for each application server instance, including the Administration Server.

On UNIX, each watchdog spawns a process for the primordial application server (`appservd`) process, which in turn spawns the `appservd` process that accepts requests. Since it starts the server, the watchdog process ID is shown in the `pid` log file in `instance_dir/logs`.

---

### NOTE

`appservd` processes on UNIX Platforms: Although you will notice that there is a single `appservd` process started for each application server instance on Windows, there are two `appservd` processes started per application server instance on UNIX systems.

On UNIX, one `appservd` process is referred to as the “primordial” process while the second `appservd` process is referred to as the “worker” process. The worker process is the process that carries out the actual processing of application requests while the primordial process acts as an overarching controller. In a future release of the application server, you will have the option to define the number of worker processes for each application server instance. In the initial release of the product, only one worker process is supported per application server instance.

---

## Adding an Application Server Instance

To add an application server instance using the Administration interface:

1. Access the Administration interface and click App Server Instances in the left pane.
2. Click the General tab.
3. On the Sun Java System Application Server Instances page, click New.
4. On the Create New Instance page, provide an instance name and a port number.

The instance name must be unique for this Administration Server and domain. The port number must not be used by any other process on the machine.

If you are using UNIX, you can also specify a UNIX user for the instance to run as.

5. Click OK.

For more information, see the online help.

To add another application server instance using the command-line interface, use the `asadmin` utility's `create-instance` command, which has the following syntax:

```
asadmin create-instance [--user admin_user] [--password admin_password] [--host  
host] [--port port] [--sysuser sys_user] [--domain domain_name] [--local=true/false]  
[--passwordfile file_name] [--secure | -s] --instanceport instance_port instance_name
```

This command has a `local` option which you can use to restart the server instance without going through the Administration Server. The `sysuser` option is only for UNIX.

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Deleting an Application Server Instance

You can delete an application server instance from your administrative domain. Be sure that you don't need the application server instance anymore before you delete it, since this process cannot be undone.

To delete an application server instance from your machine using the Administration interface:

1. Access the Administration interface and click the name of the application server instance you want to remove.
2. Click the General tab.
3. Click Delete.

For more information, see the online help.

To delete an application server instance from your machine using the command-line interface, use the `asadmin` utility's `delete-instance` command, which has the following syntax:

```
asadmin delete-instance [--user admin_user] [--password admin_password] [--host
admin_host] [--port admin_port] [--domain domain_name] [--local=true/false]
[--passwordfile file_name] [--secure | -s] instance_name
```

This command has a `local` option which you can use to delete the server instance without going through the Administration Server.

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Applying Changes to an Application Server Instance

When you change configuration information using the Administration interface or the command-line interface, the changes are not applied immediately, but are saved into special files, located in `server_instance/config/backup`. The Administration interface and command-line interface display configuration values stored in files in the above directory. Until the changes you make are applied they do not take effect. Applying changes is also called reconfiguring the server. When you apply your changes, all changes made to the configuration since the last time you applied changes take effect. Note that restarting the instance does not apply the changes automatically.

If you have made changes to the server instance configuration that require you to apply changes, a yellow icon appears next to the application server instance in the left pane's tree view, in the banner when you access the server instance, and on the server instance's main page.

**Figure 4-1** Warning Icon

To apply changes to an application server instance using the Administration interface:

1. Access the Administration interface and click the name of the application server instance you want to reconfigure.
2. Click the General tab.
3. Click Apply Changes.

When the changes are applied, the screen displays a message.

To reconfigure an application server instance using the command-line interface, use the `asadmin` utility's `reconfig` command, which has the following syntax:

```
asadmin reconfig --user admin_user [--password admin_password] [--host admin_host]
[--port admin_port] [--passwordfile file_name] [--secure | -s]
[--discardmanualchanges=true/false | --keepmanualchanges=true/false] instance_name
```

If you have made manual changes to the configuration files by editing them by hand, you must use `keepmanualchanges=true` to keep those edits during the reconfiguration (the option defaults to `false`). If you set `discardmanualchanges=true`, you discard any changes made manually. Setting `discardmanualchanges=false` (the default) does not mean the same thing as `keepmanualchanges=true`. Instead, setting it to `false` is the equivalent of not specifying the `discardmanualchanges` option.

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

For some properties, you need to restart the server, as well as apply changes, in order for your changes to take effect. These properties include all properties set in the configuration files `init.conf` and `obj.conf`, and some properties in `server.xml`. For information about these files, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

Your server will warn you if the changes require restart by putting a yellow warning icon next to the server instance left pane's tree view, in the banner when you access the server instance, and on the server instance's main page. The messages in the banner and on the page indicate that a restart is required. Once you restart the server instance, the yellow warning icons disappear.

The `server.xml` settings that do *not* require a restart include the following:

- Deploying, undeploying, and redeploying J2EE applications (EAR files), EJB modules (JAR files), web modules (WAR files), connectors (RAR files). Note that these settings also don't require Apply Changes.
- Enabling and disabling J2EE applications (EAR files), EJB modules (JAR files), web modules (WAR files), and connectors (RAR files).
- Creating, updating, and deleting resources.
- Setting monitoring enabled to true/false for the EJB container or MDB container.
- Changes to HTTP and web-container features (that is, in `server.xml`, changes to `http-service` and `web-container` and their sub-elements).

## Viewing Application Server Instance Status

You can view whether a server is started or stopped, as well as basic application server instance settings, using the Administration interface.

To view application server instance status:

1. In the left pane, click the application server instance name.
2. In the right pane, click the General tab.

You see whether the server is running or not running, as well as the hostname, port number, installation directory, and the version of the Sun Java System Application Server software.

To view the application server instance's status using the command-line interface, use the `asadmin` utility's `show-instance-status` command. The status is starting, started, stopping, or stopped. The command has the following syntax:

```
asadmin show-instance-status --user admin_user [--password admin_password]
[--host admin_host] [--port admin_port] [--passwordfile file_name] [--secure | -s]
instance_name
```

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

# Configuring JVM Settings

You can configure Java Virtual Machine (JVM) settings for your application server instance. These settings include the location of your Java home, compiler options, debugging options, and profiler information. One reason to configure these settings is to improve performance. For more information on performance see the *Sun Java System Application Server Performance and Tuning Guide*.

This section describes the following topics:

- [Configuring General Settings](#)
- [Configuring Path Settings](#)
- [Configuring JVM Options](#)
- [Configuring the JVM Profiler](#)
- [Configuring JVM Settings Using the Command-Line Interface](#)

## Configuring General Settings

To configure the JVM's general options in the Administration interface:

1. In the left pane, click the application server instance name.
2. In the right pane, click the JVM tab.
3. Click General.
4. Set the Java Home.

The Java Home is the path to the directory where the Java Developer's Kit (JDK) is installed. Sun Java System Application Server supports the Sun JDK 1.4.0\_02 or higher.

5. Choose whether to enable debugging and set debug options.

A list of debug options is available at

<http://java.sun.com/products/jpda/doc/conninv.html#Invocation>

6. Choose rmic options.

The rmic options field shows the rmic options passed to the RMI compiler at application deployment time. The `-keepgenerated` option saves generated source for stubs and ties. For more information about the `rmic` command, see <http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/rmic.html>

7. Click Save.

## Configuring Path Settings

To configure the JVM's path settings in the Administration interface:

1. In the left pane, click the application server instance name.
2. In the right pane, click the JVM tab.
3. Click Path Settings.
4. Choose a suffix for the system's classpath.
5. Choose whether to ignore the environment classpath.

If you do not ignore the classpath, the CLASSPATH environment variable is read and appended to the Sun Java System Application Server classpath. The CLASSPATH environment variable is added after the classpath-suffix, at the very end.

For a development environment, the classpath should be used. For a production environment, this classpath should be ignored to prevent environment variable side effects.

6. Set a native library path prefix and suffix.

The native library path is the automatically constructed concatenation of the Sun Java System Application Server installation relative path for its native shared libraries, the standard JRE native library path, the shell environment setting (LD\_LIBRARY\_PATH on UNIX), and any path specified in the profiler element. Since this is synthesized, it does not appear explicitly in the server configuration.

7. Click Save.

## Configuring JVM Options

To set JVM command-line options in the Administration interface:

1. In the left pane, click the application server instance name.
2. In the right pane, click the JVM tab.
3. Click JVM Options.

4. To add a JVM option, type it in the text field at the top of the screen and click Add.
5. To delete a JVM option, click the checkbox next to it and click Delete.
6. To edit a JVM option, edit the text in the JVM Option field and click Save.

For information about specific JVM options, see  
<http://java.sun.com/docs/hotspot/VMOptions.html>.

## Configuring the JVM Profiler

To configure the JVM Profiler in the Administration interface:

1. In the left pane, click the application server instance name.
2. In the right pane, click the JVM tab.
3. Click Profiler.
4. Specify the name of the profiler, its classpath and native library path, and whether it is enabled.
5. To add a JVM option for the profiler, type it in the text field at the top of the screen and click Add.
6. To delete a JVM option for the profiler, click the checkbox next to it and click Delete.
7. To edit a JVM option for the profiler, edit the text in the JVM Option field and click Save.

For more information about profilers, see the *Sun Java System Application Server Developer's Guide*.

## Configuring JVM Settings Using the Command-Line Interface

To configure JVM Settings using the command-line interface's `asadmin` utility, use the following commands:

To get all the attributes from an instance:

```
asadmin> get server_instance.java-config.*
```



To get an attribute called `classpathprefix` in `server1`:

```
asadmin> get server1.java-config.classpathprefix
```

To set an attribute called `classpathprefix` in `server1`:

```
asadmin> set server1.java-config.classpathprefix=com.sun
```

The above examples all assume you have already set the user, password, host, and port in your environment variables. For a full list of attributes, see [Appendix A, “Using the Command Line Interface.”](#)

To set JVM options using the command-line interface’s `asadmin` utility, use the following commands:

```
asadmin> create-jvm-options --user admin_user [--password admin_password]
[--host host] [--port port] [--secure | -s] [--instance instance_name]
[--profiler=true/false]
(jvm_option_name=jvm_option_value)[:jvm_option_name=jvm_option_name]*
```

```
asadmin> delete-jvm-options --user admin_user [--password admin_password]
[--host host] [--port port] [--secure | -s] [--instance instance_name]
[--profiler=true/false]
(jvm_option_name=jvm_option_value)[:jvm_option_name=jvm_option_name]*
```

Note: you can enter more than one JVM option separated by colon. If the options are used by the profiler, set `--profiler` to `true`.

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Configuring Logging Setting and Monitoring Settings

The settings on the Logging and Monitoring tabs are Logging and Monitoring settings are covered in separate chapters. For information on logging, see [Chapter 5, “Using Logging.”](#) For information on Monitoring and SNMP settings, see [Chapter 6, “Monitoring Sun Java System Application Server.”](#)

# Changing Application Server Instance Advanced Settings

An application server instance has additional settings showing the instance's locale (which determines settings such as the character set and language), the path to the server's log files, the path to the directory for deployed applications, and the path to the session store directory where passivated beans and persistent HTTP sessions are stored.

In addition, you enable application reloading and poll interval for how often to reload. Dynamic application reloading automatically checks applications for changes, and serves the updated version automatically if they have been changed. In general, you should enable dynamic reloading in a development environment and not in a production environment. The poll interval specifies the interval at which the Sun Java System Application Server checks the applications for updates.

To change the application server instance's settings using the Administration interface:

1. In the left pane, click the application server instance name.
2. On the application server instance's page, click the Advanced tab.
3. Enter the desired value in the fields.
4. Click Save.

To change the server instance's advanced setting using the command-line interface's `asadmin` utility, you use the `get` and `set` commands. When you get all the attributes for a server instance

To get all the attributes from an instance:

```
asadmin get instance_name.*
```

For example:

```
asadmin get server1.*
```

To get an attribute called `logRoot` for `server1`:

```
asadmin get server1.logRoot
```

To set an attribute called `logRoot` for `server1`:

```
asadmin set server1.logRoot=/space/log
```

The above examples all assume you have already set the user, password, host, and port in your environment variables. For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)



# Using Logging

This chapter describes the logging features and functions of the Sun Java System Application Server. In addition, components for which logging may be used are discussed.

This chapter includes the following topics:

- [About Logging](#)
- [Logging on the UNIX and Windows Platform](#)
- [Using Log Levels](#)
- [About Virtual Servers and Logging](#)
- [About Loggers](#)
- [About Client Side Logging](#)
- [Redirecting Application and Server Log Output](#)
- [Log File Management](#)
- [Configuring Logging Through the Command-line Interface](#)
- [Configuring Logging Through the Administration Interface](#)
- [Configuring the Directives for Error Logging](#)
- [Viewing the Access Log File](#)
- [Viewing the Event Log File](#)
- [Setting Log Preferences](#)
- [Running the Log Analyzer](#)

# About Logging

Logging is a useful debugging and diagnostic tool when used in applications. It also increases the developer's productivity. The Sun Java System application server's own log output can help identify and diagnose server configuration and deployment problems.

Logging within the Sun Java System Application Server uses the Java logging API. Sun Java System Application Server collects and stores logging information in two log files, namely, `access.log` and `server.log` located in the `logs` directory. You can also direct logs to your own log file.

A logged message provides more information than simply the message itself. The additional information provided includes:

- Date and time of the event.
- Log level for the event. Appserver specified Loglevel ID or name.
- Process ID (PID). PID of the appserv process.
- (Optional) Virtual server ID (vsid). The vsid that generated the message.
- Message ID. Subsystem and a four digit integer.
- Message data.

The type and order of the additional message information varies depending upon the platform used for logging and the logging service enabled for that platform. To enable the virtual server ID for logged messages, see [“Configuring the Log Service” on page 115](#).

## Logging on the UNIX and Windows Platform

This section discusses how log files are created. In addition, this section includes the following topics:

- [Default Logging in server.log](#)
- [Changing Default Log File Location](#)
- [Logging Using syslog](#)
- [Logging Using the Windows eventlog](#)

## Default Logging in server.log

On both the UNIX and Windows platforms, the log files are created in `server.log` located in the `log` sub-directory. Logs coming from all the server components and virtual servers of an instance are collected in this single file.

The default log level for the entire server can be set. However, you can also override the default log level for a particular subsystem at the subsystem level. You can also redirect `stdout` and `stderr` to the server's event log and direct the log output to the operating system's system log. Additionally, you can direct `stdout` and `stderr` content to the server's event log. Log messages by default are sent to `stderr` in addition to the specified server log file.

Another feature available is to log the virtual server ID with the log message. This is a useful feature when multiple virtual servers are used to log messages to the same log file. You can choose to write the log messages to system log. When you do so, logging is not performed on the `server.log` file. Instead the `syslog` logging service on UNIX, or the system logging service on Windows platform is used to produce and manage logs.

You can also use the `server.xml` attributes to control the contents of this file. For details about the `server.xml` file, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

### Example of server.log

The following is an example of `server.log`.

Timestamp, Log Level, (PID vsid(optional)): messageID: message

```
[01/Aug/2002:11:39:31] INFO ( 1224): CORE1116: Sun Java System Application
Server 7 2004Q2

[01/Aug/2002:11:39:36] INFO ( 1224): CORE5076: Using [Java HotSpot(TM)
Server VM, Version 1.4.0_02-20020712] from [Sun Microsystems Inc.]

[01/Aug/2002:11:39:50] INFO ( 1224): JMS5023: JMS service successfully
started. Instance Name = domain1_server1, Home = [D:\install_7_29\img\bin].

[01/Aug/2002:11:39:53] INFO ( 1224): CIS0056: Creating TCP ServerConnection
at [EndPoint [IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]

[01/Aug/2002:11:39:53] INFO ( 1224): CIS0057: Created TCP ServerConnection
at [EndPoint [IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]

[01/Aug/2002:11:39:54] INFO ( 1224): CIS0054: Creating TCP Connection from
[-] to [EndPoint [IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]
```

## Changing Default Log File Location

While deploying an enterprise or web application, you are provided the option of pre-compiling the JSPs in the application. The log messages pertaining the JSPs that are being compiled are stored in the administration server's log file usually located at `{domain_root}/{domain_name}/admin-server/logs/server.log`. This is the default logging option.

Since all the messages are logged to the same file, exceptions or errors thrown while deploying applications using pre-compiled JSPs could get lost in the volume of messages in the common log file. When multiple applications are being deployed to multiple instances under a given domain, the log messages in the admin-server will need to be carefully scrutinized for any exceptions relevant to a particular application's JSPs.

Therefore, it is better to log messages pertaining to applications that are deployed using pre-compiled JSPs, in the server instance's `server.log` file, instead of the administrative server's `server.log` file.

To redirect the log messages to your Sun Java System Application Server instance's `server.log` file, change the path to the log file in the Administrator Interface. See [“Configuring the Log Service” on page 115](#) for more information.

## Logging Using syslog

For stable operational environments where centralized logging is required `syslog` is appropriate. For environments where log output is frequently required for diagnostics and debugging, individual server instance or virtual server logs may be more manageable.

---

**NOTE**

- All logged data for the server instance and administration server in one file may prove difficult to read and debug. It is recommended that you use the `syslog` master log file only for deployed applications that are running smoothly.
  - Logged message are intermixed with all other logs from the Solaris daemon applications.
-



By using the `syslog` log file, in conjunction with `syslogd`, and the system log daemon, you can configure the `syslog.conf` file to:

- Log messages to the appropriate system log
- Write messages to the system console
- Forward logged messages to a list of users, or forward logged messages to another `syslogd` on another host over the network

---

**NOTE** Following installation of Sun Java System Application Server, the log service element attribute for the server, `use-system-logging`, is not enabled. This means that logs by default are not directed to `syslog` on UNIX or the Windows Event Log on Windows platforms. You can direct logging to `syslog` or the Windows Event Log by enabling this attribute in the Server element of `server.xml` described in the *Sun Java System Application Server Configuration File Reference*. Before setting `use-system-logging`, see “Log File Management,” on page 107.

---

## Configuring syslog

To allow better manageability and readability, you can direct messages with less severity to be stored in a separate file by configuring `syslog.conf` located in the `/etc` directory.

### To configure syslog:

1. To direct messages with less severity to be stored in a separate file add the following command to the `syslog.conf` file in Solaris:

```
daemon.debug /var/adm/iasdebug
```

2. Provide a hang-up signal to `syslogd`. This can be done by using the following command:

```
kill -HUP <PID syslogd>
```

3. Go to the Admin Server in the Administration interface and select the Write to system log option. Save, and apply changes. Restart the Admin Server for the changes to take effect.

**An example of a configured Solaris `syslog.conf` file follows:**

```
#ident"@(#)syslog.conf1.598/12/14 SMI"/* SunOS 5.0 */
#
# Copyright (c) 1991-1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (`') names
# that match m4 reserved words. Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice/dev/sysmsg
*.err;kern.debug;mail.crit/var/adm/messages
daemon.info;daemon.err;daemon.debug;daemon.alert;daemon.crit;daemon.warnin
g    /var/adm/iaslog
daemon.debug/var/adm/iasdebug
#daemon.notice;        /var/adm/iaslognotice
#daemon.warning;       /var/adm/iaslogwarning
#daemon.alert;         /var/adm/iaslogalert
#daemon.err;           /var/adm/iaslogerr

#*.alert;kern.err;daemon.err;operator
#*.alert                root
*.emerg                 *

# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.noticeifdef(`LOGHOST', /var/log/authlog, @loghost)
```

```
mail.debug ifdef(`LOGHOST', /var/log/syslog, @loghost)
#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(`LOGHOST', ,
user.err          /dev/sysmsg
user.err          /var/adm/messages
user.alert        `root, operator'
user.emerg        *
)

```

For more information, see the `syslog.conf` man page.

Any change made to `syslog.conf` will require a restart of the Sun Java System Application Server for the change to take effect.

Since logging to `syslog` means, logs from all Sun Java System Application Servers, and other daemon applications are collected in the same file, logged messages are enhanced with the following information to identify Sun Java System Application Server-specific messages from the particular server or virtual server instance:

- Unique message ID
- Timestamp
- Instance name
- Program name (`appservd` or `appserv-wdog`)
- Process ID (PID of the `appserv` process)
- Thread ID (optional)
- Server ID

The log service can be configured for both the server instance and the virtual server instance in the `server.xml` file. The log service configuration for the virtual server instance is described in the section [“About Virtual Servers and Logging” on page 103](#). The log service configuration for the server instance is described in [“Configuring Logging Through the Administration Interface” on page 114](#).

Log levels are configurable through the Administration interface for applicable subsystems and components.

For more information on the `syslog` logging mechanism used in the UNIX operating environment, use the following `man` commands at a terminal prompt:

```
man syslog
man syslogd
man syslog.conf
```

## Example of syslog messages

The following is an example of `syslog` messages.

Timestamp, hostname [instance\_name], [subsystem], [vsid], Message ID, Loglevel, Message data

```
Jul 19 14:33:18 strange /usr/lib/nfs/lockd[164]: [ID 599441 daemon.info]
Number of servers not specified. Using default of 20.

Jul 19 14:33:20 strange ntpdate[181]: [ID 558275 daemon.notice] adjust time
server 192.18.56.149 offset 0.06702 6 sec

Jul 19 14:38:13 strange xntpd[248]: [ID 204180 daemon.info] synchronisation
lost

Jul 19 14:38:47 strange server1 appservd[374]: [ID 702911 daemon.info] INFO
( 374): CORE1116: Sun Java System Application Server 7 2004Q2

Jul 19 14:38:48 strange server1 appservd[374]: [ID 702911 daemon.info] FINE
( 374): Collecting statistics for up to 1 processes with 128 threads, 200
listen sockets, and 1000 virtual servers
```

## Logging Using the Windows eventlog

For more information on the event log mechanism used in the Windows operating environment, refer to the Windows help system index for the keywords *Event Logging*.

# Using Log Levels

This section discusses log levels and how you can assign a log level for each Sun Java System Application Server subsystem.

The following topics are described:

- [About Log Levels](#)
- [Log Levels Used for syslog Configuration](#)

## About Log Levels

The Sun Java System Application Server uses the standard JDK 1.4.2\_02 log levels for selective logging of information. In addition to the standard JDK log levels, Sun Java System Application Server has added log levels designed to map more intuitively to `server.log` and to tightly integrate with Solaris.

When logged messages are routed to `server.log`, they are also mapped to log levels as defined in the “[Sun Java System Application Server Log Levels Mapped to server.log](#)” on page 103.

---

**NOTE** The default log level for the `server.log` file (or for `syslog`) for the Admin Server and the default application server instance is INFO. When the default log level is used for the application server instance, error and information messages are logged. To avoid logging such messages, change the log level to WARNING or SEVERE in the `server.xml` file or in the Administration interface of the Admin Server and the server instance.

Default log level across the server can be set in the log-service element. This affects any element that has log level set to “default”.

---

You can assign a log level for each Sun Java System Application Server subsystem for which logging is enabled. A log level is useful to streamline the amount of message information that is recorded during runtime. The level is specified in the `server.xml` file for the intended subsystem. You can specify the log level from the Administration interface for the selected subsystem, or, you can edit the `server.xml` file directly to set the desired log level for the selected subsystem.

---

**CAUTION** Manual edits to the `server.xml` file may introduce syntax errors that result in server startup failures. Guidelines for manually editing configuration files are discussed in the *Sun Java System Application Server Administrator's Configuration File Reference*, section “Manually Editing Configuration Files”.

---

To set the log level for each subsystem or component directly in the `server.xml` file, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

The Log levels described in the table “Log Levels” on page 102 match the requirements of the JDK1.4.2\_02 logging API specification. However, the log levels ALERT and FATAL are specific to Sun Java System Application Server and are not implemented in the JDK1.4.2\_02 logging API.

The following table defines the log levels and messages in Sun Java System Application Server, in increasing order of severity. The left column lists the log level designation in the Sun Java System Application Server and the right column provides a brief description of each log level.

Table 5-1 Log Levels	
Log level	Description
FINEST FINER FINE	Messages indicate extent of verbosity of debug messages. FINEST gives the maximum verbosity.
CONFIG	Messages relate to a variety of static configuration information, to assist in debugging problems that may be associated with particular configurations.
INFO	Messages are informative in nature, usually related to server configuration or server status. These messages do not indicate errors that need immediate action.  For example, a message could be logged that configuration change notification has been received; creating a new topic on MessageBroker
WARNING	Messages indicate a warning. The message would probably be accompanied by an exception.
SEVERE	Messages indicate an event of considerable importance that is likely to prevent normal application execution
ALERT*	Messages alert the user to take specific action.
FATAL*	Messages indicate a fatal error, after which server execution is not recommended. Ideally, this would be the last message before a server crash.
* Log levels specific to Sun Java System Application Server.	

<b>NOTE</b>	All messages with a log level less than INFO (FINEST, FINER, FINE, and CONFIG) provide information that help with issues relating to debugging and must be enabled as advised by technical support. Messages with a log level less than INFO are generally not localized.
-------------	---

## Log Levels Used for syslog Configuration

The following table contains a list of log levels that can be configured within the Sun Java System Application Server when using `syslog`. The left column lists the log level designation in the Sun Java System Application Server, and right column provides the corresponding log level in the `syslog` facility.

**Table 5-2** Sun Java System Application Server Log Levels Mapped to `server.log`

Sun Java System Application Server	syslog level
FINEST	LOG_DEBUG
FINER	LOG_DEBUG
FINE	LOG_DEBUG
CONFIG	LOG_INFO
INFO (default)	LOG_INFO
WARNING	LOG_WARNING
SEVERE	LOG_ERR
ALERT	LOG_ALERT
FATAL	LOG_CRIT

## About Virtual Servers and Logging

The Sun Java System Application Server can have virtual server instances. Each virtual server within a application server instance has its own identity and may have its own log file. The use of separate log files for each virtual server can help track server activity for particular transactions and resources.

To specify the log file name for a virtual server from the Administration interface, go to the HTTP Server link from the directory tree and open the server instance element under the virtual server folder to display the General tab on the right frame. You can enter the path and name of the log file for this virtual server in the Log File field.

---

**NOTE** When logging is enabled and an application is running, logged messages from the application are logged without the virtual server ID.

---

You can also direct logged messages from multiple virtual servers to one server log file. When so doing, you may wish to enable the `log-virtual-server-id` in the `log` service element of the `server.xml` file. This helps users to distinguish log messages originating from different virtual servers.

```
<log-service level="FINEST" log-stdout="false" log-stderr="false"
echo-log-messages-to-stderr="false" create-console="false"
log-virtual-server-id="true" use-system-logging="false">

</log-service>

<http-listener>

  <virtual-server-class>

    <virtual-server id="server1" http-listeners="http-listener-1"
hosts="strange" mime="mime1" state="on" accept-language="false"/>

    <virtual-server id="server2" hosts="strange" mime="mime1"/>

  </virtual-server-class>

</http-listener>
```

In this example, `<log-service log-virtual-server-id="true">` is responsible for including `virtual_server_id` in every log message. This allows you to differentiate messages coming from different virtual servers. The absence of attribute “`log-file`” in the `virtual-server` element, causes all the virtual servers to log messages to a single file.

# About Loggers

Logging can be enabled or disabled selectively at the subsystem level. Logging control for each subsystem is specified in the `server.xml` file, as described in the *Sun Java System Application Server Configuration File Reference*. Each subsystem has its own logger in accordance with the requirements of the JDK1.4.2\_02 logging API.

In the following table, the left column defines the subsystem and the right column the element in the `server.xml` file for each subsystem.

**Table 5-3** Subsystems and Locations in Sun Java System Application Server

Subsystem	Element
Administration Server	<admin-service>
EJB Container	<ejb-container>
Web Container	<web-container>



**Table 5-3** Subsystems and Locations in Sun Java System Application Server

Subsystem	Element
MDB Container	<mdb-container>
Sun Java System Message Queue (JMS service)	<jms-service>
Security Service	<security-service>
Java Transaction Service (JTS)	<transaction-service>
Object Request Broker (ORB)	<iiop-service>
Default handlers <sup>1</sup>	<log-service>

1. Default handlers refers to the default logger associated with all `server.xml` entries that are not associated with a particular subsystem, such as the utility classes.

---

**NOTE** On the Windows platform, if you choose to send logs to the Windows `server.log`, only messages with level INFO, WARNING, SEVERE, ALERT or FATAL are logged to Windows Event Log.

---

The table “[Log Levels](#)” on [page 102](#) defines the log levels, in increasing order of severity, provided for messages in Sun Java System Application Server. These log levels match the requirements of the JDK1.4.2\_02 logging API specification. In addition, the log levels ALERT and FATAL are specific to Sun Java System Application Server and are not supported in the JDK1.4.2\_02 logging API.

## About Client Side Logging

The Application Client Container (ACC) has its own log service and can only log to a local file.

The ACC typically runs in its own process, on a different host from the application server. As such it has its own logging infrastructure and its own log file. The ACC configuration is held in the file `sun-acc.xml`.

The client subsystem logging element for the ACC is `log-service`. The following table lists the element, and the attributes, each with the noted default and range of values.

**Table 5-4** ACC Logging Element

Element	Attribute	Description
log-service	file	ACC log file; when empty or missing, log to stdout.
log-service	level	ACC log level.

An example of the `sun-acc.xml` file is provided in the *Sun Java System Application Server Configuration File Reference*.

## Redirecting Application and Server Log Output

For developers, it is important that the application logs and server logs be made readily available during unit testing for application components and J2EE applications. On the Windows platform, developers prefer to see server log messages displayed in a command window on the desktop. On the UNIX platform, many developers are comfortable with simply having the log messages stream to `stderr` in the terminal window in which the server instance is started, or, use the command `tail -f` to see the log messages written in log files.

The `server.xml` file contains attributes that can be set for `stdout` and `stderr` to direct logged messages to a log file or to the terminal window, and so forth. See the *Sun Java System Application Server Configuration File Reference* for more information on the use of `stdout` and `stderr`.

See [“Configuring the Log Service” on page 115](#) for information on the log service.

## Log File Management

You can set up your access and event log (`server.log`) files to be automatically archived. At a certain time, or after a specified interval, your logs are rotated. Sun Java System Application Server saves the old log files and stamps the saved file with a name that includes the date and time they were saved.

---

**NOTE** Although you can create multiple virtual servers and associate a log file for each virtual server, log rotation settings for individual virtual servers is not supported.

---

For example, you can set up your access log files to rotate every hour, and Sun Java System Application Server saves and names the file “access.199907152400,” where name of the log file, year, month, day, and 24-hour time is concatenated together into a single character string. The exact format of the log archive file varies depending upon which type of log rotation you set up.

---

**NOTE** These facilities are primarily provided for non-Solaris platforms.

For Solaris, these facilities are not enabled by default, and you must use the native Solaris operating system log management facilities such as `logadm` on Solaris 9. On Solaris 8, the preferred utility for log management would be the `cron` facility described in [“Using the Solaris cron Utility to Schedule Execution of logadm” on page 113](#).

---

Depending on the operating system, there are four distinct ways you can perform log rotations. These are discussed in the section that follows. Topics include:

For UNIX and Windows:

- [Internal-daemon Log Rotation](#)
- [Scheduler Based Log Rotation](#)

For Solaris 9

- Using Solaris `logadm` utility. For more information, see [“Rotation Using Solaris logadm Utility” on page 109](#).

For Solaris (any version)

- Using Solaris `cron` utility. For more information, see [“Rotation Using Solaris “cron” Utility” on page 111](#).

## Internal-daemon Log Rotation

Internal-daemon log rotation is available for both UNIX and Windows operating systems.

Internal-daemon log rotation occurs within the HTTP daemon and can only be configured at the server instance startup time. Internal daemon log rotation allows the server to rotate logs internally without requiring a server restart.

Logs rotated using this method are saved in the following format:

```
access.<YYYY><MM><DD><HHMM>
```

```
error. <YYYY><MM><DD><HHMM>
```

You can specify the time used as a basis to rotate log files and start a new log file. For example, if the rotation start time is 12:00 a.m., and the rotation interval is 1440 minutes (one day), a new log file is created immediately upon save regardless of the present time and collects information until the rotation start time. The log file rotates every day at 12:00 a.m., and the access log is stamped at 12:00 a.m. and saved as `access.199907152400`. Likewise, if you set the interval at 240 minutes (four hours), the four-hour intervals begin at 12:00 a.m. such that the access log files will contain information gathered from 12:00 a.m. to 4:00 a.m., from 4:00 a.m. to 8:00 a.m., and so forth.

If log rotation is enabled, log file rotation starts at server startup. The first log file to be rotated gathers information from the current time until the next rotation time. Using the previous example, if you set your start time at 12:00 a.m. and your rotation interval at 240 minutes, and the current time is 6:00 a.m., the first log file to be rotated will contain the information gathered from 6:00 a.m. to 8:00 a.m, and the next log file will contain information from 8:00 a.m. to 12:00 p.m. (noon), and so forth.

## Scheduler Based Log Rotation

Scheduler log rotation, allows you to archive log files immediately or have the server archive log files at a specific time on specific days. To archive log files immediately select Admin Server from the left pane of the Administration interface. Then, click on the Logging link at the top of the right page. Next, click Log Rotation. Finally, click Archive.

Logs rotated using the scheduler method are saved as the original filename followed by the date and time the file was rotated. For example, `access` might become `access.24Apr-0430PM` when it is rotated at 4:30 p.m.

Log rotation is initialized at server startup. If rotation is turned on, Sun Java System Application Server creates a time-stamped access log file and rotation starts at server startup.

Once the rotation starts, Sun Java System Application Server creates a new time stamped log file when there is a request or error that needs to be logged to the access or error log file and it occurs after the prior-scheduled “next rotate time”.

---

**NOTE** For the Windows platform, and for server logging directed to a file other than `syslog` on Solaris, you must archive the server logs.

---

To archive log files and to specify use of the `schedulerd` control method, select Admin Server from the left pane of the Administration interface. Then, click on the Logging link at the top of the right page. Next, click the Scheduler based Log Rotation box. Finally, OK. The current state of the `scheduler` is indicated.

## Rotation Using Solaris `logadm` Utility

The Solaris 9 operating system incorporates the utility `logadm` which can be used to perform an array of functions with logged messages.

Particular to the Sun Java System Application Server, this utility is useful for performing log rotation tasks when running it from the Solaris cron utility, described in [“Using the Solaris cron Utility to Schedule Execution of `logadm`” on page 113](#).

You can specify the following log rotation details with respect to log files:

- All log file names on the system that must be rotated
- Rotation interval
- Condition that will trigger the rotation
- Number of backup log files to be saved
- Naming convention of backup log files to be saved

The above details are specified in the file `logadm.conf` located in:

```
n /etc/logadm.conf
```

A sample `logadm.conf` file follows:

```
# Copyright 2001-2002 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)logadm.conf 1.2 02/02/13 SMI"
#
# logadm.conf
```

```

#
# Default settings for system log file management.
# The -w option to logadm(1M) is the preferred way to write to this
# file,
# but if you do edit it by hand, use "logadm -V" to check it for
# errors.
#
# The format of lines in this file is:
# <logname> <:options>
# For each logname listed here, the default options to logadm
# are given. Options given on the logadm command line override
# the defaults contained in this file.
# # logadm typically runs early every morning via an entry in
# root's crontab (see crontab(1)).
#
/var/log/syslog -C 8 -P 'Tue Jul 9 10:10:00 2002' -a 'kill -HUP `cat
/var/run/syslog.pid`' /var/adm/messages -C 4 -P 'Tue Jul 30 10:10:00 2002'
-a
'kill -HUP `cat /var/run/syslog.pid`' /var/cron/log -c -s 512k -t
/var/cron/olog
/var/lp/logs/lpsched -C 2 -N -t '$file.$N'
#
# The entry below is used by turnacct(1M)
#
/var/adm/pacct -C 0 -N -a '/usr/lib/acct/accton pacct' -g adm -m 664 -o adm
-p never
#
# The entry below will rotate Sun Java System application server's default
logfile
# every day provided the current logfile size is >= 512k. It will compress
# the old log file before archiving it and also delete the old files after
30

```

```
# days. The compression is done with gzip(1) and the resulting log file has
# the suffix of .gz.
```

```
/var/appserver/domains/domain1/server1/logs/server.log -A 30d -s 512k -p 1d
-z
```

Alternatively, you can start log rotation on a specific file by invoking the `logadm` command interactively.

The following example rotates `syslog` and keeps eight log files. Old log files are placed in the directory `/var/oldlogs` instead of `/var/log`:

```
% logadm -C8 -t '/var/oldlogs/syslog.$n' /var/log/syslog
```

You can also use an interactive command-line option to invoke rotation on a file specified in `/etc/logadm.conf`, but with different or modified options.

If options are specified both in `/etc/logadm.conf` and on the command-line, those in the `/etc/logadm.conf` file are applied first. Therefore, the command-line options override those in `/etc/logadm.conf`. An example of this is as follows:

```
% logadm /var/appserver/domains/domain1/server1/logs/server.log -p now
```

The above command rotates the given file using all the options provided for that file in `/etc/logadm.conf`.

---

<b>NOTE</b>	When multiple options are specified at a time, there is an implied <b>AND</b> between them. This means that all conditions must be met before the log is rotated.
-------------	---

---

For detailed information on `logadm` utility and its options, please refer to its manpage as follows:

```
% man logadm
```

OR

```
% logadm -h
```

## Rotation Using Solaris “cron” Utility

On Solaris 8, the `cron` utility can be used to perform application server log rotations. Cron-based log rotation is initialized at server startup. If rotation is turned on, the server creates a time-stamped access log file and rotation starts at server startup.

To perform application server log rotation use the following command:

```
% crontab -e
```

This starts your favorite editor (defined by `env.` variable `$EDITOR`) so that you can provide the list of cron entries.

---

**NOTE** This command also invokes `/etc/cron.d/logchecker` script as soon you exit from the editor. This script feeds the changed/new crontab entries to cron daemon. Therefore, entries added this way are immediately picked up by cron daemon and log rotations starts immediately.

You need not restart the cron daemon to enable log rotations.

---

This section includes the following topics:

- [About the crontab Entry Format](#)
- [Using the Solaris cron Utility to Schedule Execution of logadm](#)

### About the crontab Entry Format

A `crontab` file consists of lines of six fields each. The fields are separated by spaces or tabs. The first five fields are integer patterns that specify the following:

- minute (0-59)
- hour (0-23),
- day of the month (1-31),
- month of the year (1-12),
- day of the week (0-6 with 0=Sunday).

Using this format, you can specify access and eventlog files to be rotated at specified time of day/week/month and schedule to repeat the rotations. For example,

```
0 0 * * 1-5
```

```
/opt/SUNWappserver7/appserver/domains/domain1/server1/bin/rotatelog
```

```
0 12 * * 1-5
```

```
/opt/SUNWappserver7/appserver/domains/domain1/server1/bin/rotatelog
```

```
0 * * * 1-5
```



```
/opt/SUNWappserver7/appserver/domains/domain1/mainserver/bin/rotatelogs
```

This rotates access and log files for server1 at midnight and noon every day between Monday and Friday; while access and log files for mainserver is rotated every hour of every day between Monday and Friday.

The crontab files are stored under `/var/spool/cron/crontabs/`. You can create the crontab file as an end user or root. Depending on your privileges, you can see the crontab entries by using the following command:

```
% crontab -l username
```

### Using the Solaris cron Utility to Schedule Execution of logadm

The `cron` command starts a process that executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in crontab files in the directory `/var/spool/cron/crontabs/`.

As an example of a regularly scheduled command used with `cron`, the following entry in crontab will start `logadm` everyday at midnight.

```
0 0 * * 0-6 logadm
```

Note that users can submit their own crontab file using the `crontab(1)` command.

To keep a log of all actions taken by `cron`, `CRONLOG=YES` (by default) must be specified in the `/etc/default/cron` file. `/etc/cron.d/logchecker` is a script that checks to see if the log file has exceeded the system `ulimit`. If so, the log file is moved to `/var/cron/olog`.

## Configuring Logging Through the Command-line Interface

You can configure aspects of the logging service from the command-line for the server instance and for the virtual server instance.

---

**NOTE** All the command examples in this section assume that the environment variables have been set.

---

To get all of the log-service attributes for the server instance:

```
asadmin> get instance_name.log-service.*
```

The log-service attributes are also defined in the table [“Log Service Attributes” on page 116](#).

An example of using this command with a designated server instance name follows:

```
asadmin> get server1.log-service.*
```

The list of attributes for the logging service of the *server1* instance is returned. You can configure each listed attribute by using the `set` command.

To enable the logging of a virtual server ID for the virtual server instance, enter the following command at the terminal prompt:

```
asadmin> get instance_name.LogVirtualServerId
```

The current state of the `LogVirtualServerId` is returned. If the state is false, you can enable it with the `set` command as follows:

```
asadmin> set instance_name.LogVirtualServerId=true
```

To set the log file name for a virtual server instance, use the `set` command as follows:

```
asadmin> set instance_name.virtual-server.<virtual server id>.logFile=<log file>
```

As an example, the following `set log file` command is issued:

```
asadmin> set
instance2.virtual-server.instance2.logFile=/space/IAS7se/appserver7/appserv/domain1/domain1/instance2/logs/log
```

For more information on command syntax, see the command-line interface help.

For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Configuring Logging Through the Administration Interface

This section describes the tasks you can perform through the Sun Java System Application Server Administration interface to configure the available logging service options for both server-wide (global) elements, directives, and application components.

This section includes the following topics:

- [Configuring the Log Service](#)
- [Configuring Logging for Application Server Components and Subsystems](#)
- [Configuring the Directives for Error Logging](#)

## Configuring the Log Service

The log service is an element within the J2EE Service Element category in the `server.xml` file, as described in the *Sun Java System Application Server Configuration File Reference*. The log service is used to configure the system logging service, which includes the following log files:

- Server log
- Access log
- Transaction log
- Virtual server log

Configuration of the system logging service includes specifying values for attributes of the log service element.

You can configure the following attributes for the log service element through the Administration interface.

- Log File
- Default Log Level
- Log Standard Out content to event log
- Log Standard Error content to event log
- Echo to Standard Error
- Create Console
- Log Virtual Server ID
- Write to System Log

The Log Service link is accessible from the expanded tree hierarchy for the server instance in the left pane of the Administration interface. The following table describes each of the attributes that can be configured, along with the default and range of values allowed.

**Table 5-5** Log Service Attributes

Attribute	Default	Description
file	server.log <sup>1</sup>	(optional) Overrides the name or location of the server log. The file and directory in which the server log is kept must be writable by whatever user account used to run the server.
level	INFO	(optional) Controls the default type of messages logged by other elements to the server log. Allowed values are as follows, from highest to lowest: FINEST, FINER, FINE, CONFIG, INFO, WARNING, SEVERE, ALERT, FATAL.  Each value logs all messages for all lower values; for example, FINEST logs all messages, and FATAL logs only FATAL messages. The default value is INFO, which logs all INFO, WARNING, SEVERE, ALERT, and FATAL messages.
log-stdout	True	(optional) If true, redirects stdout output to the server log. Legal values are on, off, yes, no, 1, 0, true, false.
log-stderr	True	(optional) If true, redirects stderr output to the server log. Legal values are on, off, yes, no, 1, 0, true, false.
echo-log-messages-to-stderr	True	(optional) If true, sends log messages to stderr in addition to the server log. Legal values are on, off, yes, no, 1, 0, true, false.
create-console	False	(optional) If true, creates a console window on Windows operating system for stderr output. Legal values are on, off, yes, no, 1, 0, true, false.
log-virtual-server-id	False	(optional) If true, virtual server IDs are displayed in the virtual server logs. These are useful if multiple virtual-server elements share the same log file.
use-system-log	False	If true, uses the UNIX syslog service or Windows Event Logging to produce and manage logs.

1. In the directory specified by the log-root attribute of the server element.

## Configuring Logging for Application Server Components and Subsystems

This section describes how to enable logging and select a log level for Sun Java System Application Server components and subsystems. Note that the Java Transaction Service component has more than one log file. Since most of the components and subsystems are handled in the same way with respect to configuring a log level, the procedure to select a log level is documented one time only, for the indicated grouping of components and subsystems.

The following components and subsystems can utilize selective logging of server messages. You can become familiar with the components and subsystems from the references to other topics within this guide as indicated.

- ORB - Configuring Support for Corba-based Clients
- Web Container - Configuring J2EE Services
- EJB Container - Configuring J2EE Services
- MDB Container - Configuring J2EE Services (within the EJB container)
- Java Transaction Service - Configuring J2EE Services
- JMS Service - Java Message Service
- Virtual Server - Using Virtual Servers

### To Specify a Log Level

To specify a log level for ORB, Web Container, EJB Container, MDB Container (within the EJB Container), Java Transaction Service, and JMS Service, perform the following procedure:

1. In the left pane of the Administration interface, expand the Sun Java System Application Server instance to display the components and subsystems you wish to edit.
2. Click the link for the desired component or subsystem.
3. In the right page of the Administration interface, select one of the following log level parameters, from the Log Level drop-down list. The log levels are described in [“About Log Levels” on page 101](#).

### To Specify a Log File: (Virtual Server)

1. In the left pane of the Administration interface, expand the Sun Java System Application Server instance to display the HTTP Server subsystem.

2. Click the HTTP Server link.
3. Click the Virtual Server link.
4. Click the desired server instance link.
5. In the right pane of the Administration interface, under the General tab, enter the desired directory path and file name for the Log File field.

### To Specify a Transaction Log Location: (Java Transaction Service)

1. In the left pane of the Administration interface, expand the Sun Java System Application Server instance to display the Transaction Service subsystem.
2. Click the Transaction Service link.
3. In the right pane of the Administration interface, under the Advanced field group, enter the desired directory path and file name for the Transaction Log Location field.

## Configuring the Directives for Error Logging

Sun Java System Application Server includes error logging directives for the `init.conf` file. The following directives are included:

- **Error Log Date Format.** The `ErrorLogDateFormat` directive specifies the date format that the server logs use.
- **Log Flush Interval.** The `LogFlushInterval` determines the maximum time interval, in seconds, before which the access log is flushed from the memory into the `access.log` file.
- **Pid Log.** `PidLog` specifies a file in which to record the process ID (pid) of the base server process. Some of the server support programs assume that this log is in the server root, in `logs/pid`.

All the directives for `init.conf` are fully described in the *Sun Java System Application Server Configuration File Reference*.

## Viewing the Access Log File

You can view both the Administrator server's and the Sun Java System Application Server instance's http log files.

To view the Administration Server's http log, select the Admin Server from the left pane in the Administration interface, and then choose the Logging tab from the right page. The View HTTP Access Log link is displayed. Select this link to view the configured access log.

To view an access log for the application server instance, click the server instance desired from the left pane of the Administration interface. Click on the Logging tab from the right pane. Click the View Access Log link to display the configured active access log for that server instance.

## Viewing the Event Log File

You can view both the Administrator server's and the Sun Java System Application Server instance's active event log files.

To view the Administrator server's event log, select the Admin Server from the left pane, and then choose the Logging tab from the right page. The View Event Log link will be displayed. Select this link to view the configured event log.

To view the event log for the application server instance, click the server instance desired from the left pane of the Administration interface, and then choose the Logging tab from the right pane. The View Event Log link will be displayed. Select this link to view the configured event log.

## Setting Log Preferences

During installation, an access log file named `access` is created for the server. You can customize access logging for any resource by specifying whether to log accesses, what format to use for logging, and whether the server should spend time looking up the domain names of clients when they access a resource.

To use one log file for multiple virtual servers, `LogVsId` should be turned on in the `server.xml` file for the event log. See the *Sun Java System Application Server Configuration File Reference* for details. Alternatively, `LogVsID` can be turned on in the Admin Server Logging tab of the Administration interface.

Follow the steps below to enable the Log virtual server ID from the Administration interface. The changes are effected following restart of the Administration server.

1. Click the Admin Server from the left pane of the Administration interface.
2. Click the Logging tab on the right page.

3. Click within the check box for Log virtual server ID.
4. Click the Save button to apply the changes to the Sun Java System Application Server.

This setting requires a restart of the Sun Java System Application Server for the change to take effect.

## Running the Log Analyzer

The `flexanlg` is a Log Analyzer tool used for log file reporting. The Log Analyzer can be used only when logging is directed to a file other than `syslog`.

Use the log analyzer to generate statistics about your default server, such as a summary of activity, most commonly accessed URLs, times during the day when the server is accessed most frequently, and so on. The log analyzer cannot generate statistics for virtual servers other than the default server. However, statistics can be viewed for each virtual server as described in [“Viewing the Access Log File” on page 118](#).

---

**NOTE** Before running the log analyzer, you must rotate the server logs. For more information, see [“Log File Management” on page 106](#).

---

You can run the log analyzer command from the command-line, by running the tool `flexanlg`, which is in the directory `install_dir/bin/flexanlg`.

To run `flexanlg`, type the following command and options at the command prompt:

```
flexanlg [ -P ] [-n name] [-x] [-r] [-p order] [-i file]* [ -m metafile ]*
[ o file] [ c opts] [-t opts] [-l opts] [-h help]
```

Command Options (Options marked with \* can be repeated).

`-i filename`

input log file(s)

`-P`

proxy log format

`-n servername`

the name of the server



```

-x
output in HTML

-r
resolve IP addresses to hostnames

-p [ c, t, l ]
output order; default order is counts, time statistics, and lists

-m filename
meta file(s)

-o filename
output log file; default is stdout

-c [ h, n, r, f, e, u, o, k, c, z ]
count these items; default is: h, n, r, e, u, o, k, c

    h: total hits
    n: 304 Not Modified status codes (use local copy)
    r: 302 Found status codes (redirects)
    f: 404 Not Found status codes (Document Not Found)
    e: 500 Server Error status codes (Misconfiguration)
    u: total unique URLs
    o: total unique hosts
    k: total kilobytes transferred
    c: total kilobytes saved by caches
    z: Do not count any items

-t [ sx, mx, hx, xx, z ]
find general statistics; default is: s5m5h24x10

    s (number): Find top (number) seconds of log
    m (number): Find top (number) minutes of log
    h (number): Find top (number) hours of log

```

```

u (number): Find top (number) users of log
a (number): Find top (number) user agents of log
r (number): Find top (number) referers of log
x (number): Find top (number) for miscellaneous keywords
z: Do not find any general statistics

-l [ cx, hx ]

Make a list of the specified suboption; default is: c+3h5

c (x, +x): most commonly accessed URLs
x: only list x entries
+x: only list if accessed more than x times
h (x, +x): hosts or IP addresses most often accessing your server
x: only list x entries
+x: only list if accessed more than x times
z: Do not make any lists

EXAMPLE: Using the flexanlg command

flexanlg -i /var/opt/SUNWappserver7/domains/domain1/server1/logs/access

```

---

**NOTE** Before running the log analyzer, you should archive the server logs.

---

## Viewing Events (Windows)

In addition to logging errors to the `server.log` file, Sun Java System Application Server logs severe system errors to the Event Viewer. The Event Viewer lets you monitor events on your system. Use the Event Viewer to see errors resulting from fundamental configuration problems, which can occur before the error log is opened.

To use the Event Viewer, perform the following steps:

1. From the Start menu, select Programs and then Administrative Tools. Choose Event Viewer in the Administrative Tools program group.

2. Choose Application from the Log menu.

The Application log appears in the EventViewer. Errors from Sun Java System Application Server has a source label of `https-serverid`.

3. Choose Find from the View menu to search for one of these labels in the log. Choose Refresh from the View menu to see updated log entries.

For more information about the Event Viewer, consult your system documentation.



# Monitoring Sun Java System Application Server

This chapter contains information about the monitoring and Simple Network Management Protocol (SNMP) features and functions available in Sun Java System Application Server.

This chapter includes the following sections:

- [About Monitoring Sun Java System Application Server](#)
- [Extracting Monitoring Data Using the CLI](#)
- [Administering the Transaction Service Using the CLI](#)
- [Using HTTP Quality of Service](#)
- [About SNMP](#)
- [Setting Up SNMP](#)
- [Enabling and Starting the SNMP Master Agent](#)

## About Monitoring Sun Java System Application Server

You can monitor Sun Java System Application Server by collecting activity statistics from strategic data points on your system. The statistics show you how many requests your server is handling and how well it is handling those requests. You can view some statistics for individual virtual servers and others for the entire Sun Java System application server instance. Either the `asadmin` utility or SNMP can be used to monitor the Sun Java System Application Server.

The following topics are addressed in this section:

- [Statistics](#)
- [SNMP](#)
- [HTTP Server Monitoring](#)
- [Administration Server Security Auditing](#)
- [Application Components and Subsystems Monitoring](#)
- [Quality of Service \(QOS\)](#)

## Statistics

Statistics collection is always enabled for most of the Sun Java System Application Server application components and subsystems, including the HTTP server; no enabling functions are required. However, there are some statistics that are collected only when monitoring is explicitly enabled on that subsystem, or only when relevant functionality is enabled. These statistics include the following data points:

- Statistics for EJB methods
- Active transactions
- Connections (only if Quality of Service is enabled)
- DNS (only if DNS Cache is enabled)

Monitoring can be enabled for application subsystems or components from the Administration interface, as described in “[Application Components and Subsystems Monitoring](#)” on page 127.

If the server monitor reports that the server is handling a large number of requests, you may need to adjust the server configuration or the system’s network kernel. For more information on adjusting your server’s configuration, see the *Sun Java System Application Server Performance Tuning and Sizing Guide*.

## SNMP

The Sun Java System Application Server provides network management information through its information gathering tools using Simple Network Management Protocol (SNMP), a protocol used to exchange management and monitoring information across a network. Using SNMP, programs called *agents*

monitor various devices on the network (hubs, routers, bridges, and so on). Another program collects the data from the agents. The database created by the monitoring operations is called a *management information base* (MIB). This data is used to check if all devices on the network are operating properly.

While only the HTTP server can be monitored using SNMP; all components and systems can be monitored using the command-line interface (CLI)

For more information on SNMP, see [“About SNMP” on page 161](#) and [“Setting Up SNMP” on page 169](#).

## HTTP Server Monitoring

HTTP server monitoring is enabled by default, which means it doesn't need to be specifically turned on. HTTP server monitoring is based on an XML file and is accessed using the `asadmin` command as a set of three monitorable attributes. The elements, subelements, and the attributes of this XML file are described in [“Monitorable HTTP Server Elements” on page 147](#) and [“Monitorable HTTP Server Attributes” on page 148](#).

---

<b>NOTE</b>	Only HTTP server statistics are available using SNMP. Statistics for all the subsystems of the Sun Java System Application Server, including the HTTP server, are available using the command-line interface.
-------------	---

---

For more information about using `asadmin`, see [“Using the Command Line Interface” on page 559](#).

## Application Components and Subsystems Monitoring

Some of the subsystems or components within the Sun Java System Application Server do not need to have monitoring enabled because the relevant statistics are always collected. For example, monitoring for application components such as containers can be enabled or disabled. When monitoring is enabled, in addition to the statistics that are always collected, additional statistics on all EJB methods are also collected. Monitoring for JDBC connection pools is always enabled. A connection pool is initialized upon first access and relevant statistics can be monitored anytime after that.

For a full list of monitorable data points, see [“Monitorable Attribute Names” on page 143](#).

You can enable monitoring for selected application components and subsystems from the Administration interface or from the command-line interface (CLI). For example, to enable monitoring from the CLI for the EJB container, type the following command from a terminal window:

```
set server1.ejb-container.monitoringEnabled=true
reconfig server1
```

where `server1` is the instance name.

The equivalent functionality can be accessed on the Administration interface under the Containers node.

The following topics are addressed in this section:

- [Monitoring for Container Subsystems](#)
- [Monitoring for the ORB Service](#)
- [Monitoring for the Transaction Service](#)

## Monitoring for Container Subsystems

In the case of the EJB container, when monitoring is enabled, the statistics related to the methods for all entity beans, stateful session beans, and stateless session beans are collected. These statistics include:

- Total number of errors
- Total number of calls
- Total number of successes
- Execution time, in milliseconds (for last invocation of the method)

All other statistics for container subsystems are always collected. Some of the monitored data points include statistics relative to:

- Initial, minimum, and maximum stateless beans in pool
- Minimum and preferred number of stateful and entity beans in cache
- Minimum and preferred number of stateless session beans in cache
- Number of beans created and destroyed
- Other related statistics



## Monitoring for the ORB Service

For the ORB service, monitored data points include statistics gathered for the ORB connection and the ORB thread pool. ORB statistics are always collected and, therefore, it is not necessary to enable monitoring for the ORB service.

## Monitoring for the Transaction Service

For the Java Transaction Service (JTS) service, monitored data points include:

- Total completed transactions
- Total rolled back transactions
- Total inflight transactions
- List of inflight transactions

Refer to [“Administering the Transaction Service Using the CLI”](#) on page 155 for further information.

# Administration Server Security Auditing

Sun Java System Application Server provides a built in security audit logging service to monitor events that directly modify the configuration of the server.

The audit logging is implemented within the admin module of Sun Java System Application Server. When the administration server starts up, a global audit session is created and persists until the administration server is stopped.

To monitor administration-related changes to Application Server instances, you need to perform the following tasks:

---

<b>NOTE</b>	The audit service does not monitor JVM specific configuration settings, the creation or deletion of domains, user-level ACL settings for customer applications, as well as a few other non-critical settings.
-------------	---

---

- [Configuring Administration Server to Use LDAP](#)
- [Setting Up Audit Services Configuration Files](#)
- [Viewing Security Audit Output](#)

## Configuring Administration Server to Use LDAP

This section describes the procedure to configure user authentication to Application Server administration server using the Lightweight Directory Access Protocol (LDAP).

1. Install a LDAP database and create an `admin` group, to which the Application Server administrators will belong.
2. Create one or more LDAP users (who will administer Application Server instances) and add them to the `admin` group.
3. Edit the `appserver_install_dir/domain1/admin-server/config/generated.admin-server.acl` file. Replace the contents of the file with the following:

```
version 3.0;

acl "admin-defaults";

authenticate(user, group) {

    method="basic";

    database="default";

    prompt="LDAP user/password";

};

deny with
file="appserver_install_dir/domains/domain1/admin-server/config/admin-deny
msg.html";

deny (all) user="anyone";

allow (all) (group = "{ADMIN_GROUP}") );
```

4. Replace `{ADMIN_GROUP}` with the LDAP admin group to which the admin users belong.

5. Edit the *appserver\_install\_dir*/domain1/admin-server/config/dbswitch.conf file. Replace the contents of the file with the following:

```
directory default ldap://{LDAP_SERVER}:{PORT}/{BASE_DN}
default:binddn {BIND_DN}
default:bindpw {BIND_PW}
```

For example:

```
directory default ldap://drectory.sun.com:1389/o=WebAuth
default:binddn cn=Directory Manager
default:bindpw dirmanager
```

6. Edit the *appserver\_install\_dir*/domain1/admin-server/config/obj.conf file. Comment out the following line of the default object section:

```
PathCheck fn="admin-check-admpw"

admpwfile="appserver_install_dir/domains/domain1/admin-server/config/admpw"
" admdns="*" admip="*" final="true"
```

7. Insert the following lines in the default object section, below the PathCheck fn="unix-uri-clean" line:

```
PathCheck fn="check-acl" acl="admin-defaults"
```

The updated obj.conf should look something like the following:

```
<Object name="default">

AuthTrans fn="match-browser" browser="*MSIE*"
ssl-unclean-shutdown="true"

NameTrans fn="ntrans-j2ee" name="j2ee"

NameTrans fn="admin-uri2path" root="/admin"

#PathCheck fn="admin-check-admpw"

admpwfile="/var/opt/SUNWappserver7/domains/domain1/admin-server/config/
admpw" admdns="*.iplanet.com" admip="*" final=false

PathCheck fn=find-pathinfo

PathCheck fn="unix-uri-clean"

PathCheck fn="check-acl" acl="admin-defaults"

PathCheck fn=find-index index-names="index.html"

ObjectType fn="type-by-extension"
```

```
ObjectType fn="force-type" type="magnus-internal/cgi"
Service fn="send-cgi" type="magnus-internal/cgi"
Service fn="imagemap" method="GET" type="magnus-internal/imagemap"
Service fn="send-file" method="GET"
Error fn="error-j2ee"
Error fn="admin-error" reason="server error"
AddLog fn="flex-log" name="access"
</Object>
```

## Setting Up Audit Services Configuration Files

To enable audit logging for the Administration Server, you have to create two configuration files in the following locations:

- The global configuration file: `/etc/defaults/slasaudit`
- The Application Server instance specific configuration file:  
`appserver_install_dir/slasaudit`

The location of these two files can be set by using the following environment variables:

- For the global configuration file: `slasaudit.default.file`
- For Application Server instance specific configuration file:  
`slasaudit.config.file`

Here's an example: On Solaris, use the following command:

```
setenv slasaudit.config.file /etc/opt/appserver7/auditconfig
```

Configure these two files using the procedures described in [“Audit Configuration Settings” on page 133](#).

---

<b>NOTE</b>	Settings specified in the instance configuration file will override those of the global configuration file.
-------------	---

---

## Audit Configuration Settings

The global and instance specific configuration files can contain the same settings. However, the instance specific settings will take precedence over the global configuration settings.

A sample audit configuration file for Solaris is shown here.

```
# This configuration file specifies the default audit
# type and audit session class file for this SIAS installation
#
# Syntax:
# audit.default.type={Audit_Type}
# audit.{Audit_Type}.class={Audit_Session_Class}
#
# Default audit types are: Log and Solaris
#

audit.default.type=Solaris

audit.Log.class=com.ipanet.ias.admin.audit.LogAuditSession

audit.Solaris.class=com.ipanet.ias.admin.audit.SolarisAuditSession
```

At the very minimum, the configuration files should contain a minimum of two entries:

- The default audit type, and
- The audit type definition.

The default audit type entry must match one of the two audit type definitions-Log or Solaris.

For platforms other than Solaris, set the `audit.default.type` to Log and remove the `audit.Solaris.class` line.

If the configuration file does not exist, or if a default audit type is not specified, security audit events will not be logged by the administration server.

## Viewing Security Audit Output

This section describes the output of the security audit logs and how to view them on different platforms. The following topics are described:

- [Viewing Security Audit Logs on Non-Solaris 9 Platforms](#)
- [Viewing Security Audit Logs Using BSM Auditing on Solaris](#)

## Viewing Security Audit Logs on Non-Solaris 9 Platforms

Security audit logs on non-Solaris 9 platforms, where native BSM auditing is not supported, audit events will be written to the default server log of the administration server instance.

There following three distinct audit events will be logged:

**Login Event** Login events are generated at startup of the administration server instance and indicates that an audit session has been created.

**Modify Event** Modify events correspond to all configuration changes which are audited by the administration server. Each modify event will contain the following data:

- **ObjectName**  
ObjectName specifies the configuration item that is being modified.
- **RemoteHost**  
RemoteHost specifies the remote host which initiated the request. Due to Application Server restrictions on remote administration, this value will always indicate the localhost.
- **Type**  
Type specifies the kind of change that has been made, such as, create, add, delete, and update.
- **Mode**  
Mode specifies whether the request originated from the CLI or the admin GUI.
- **Value**  
Value specifies the new or modified value of the ObjectName.

**Logout Event** Logout events are generated at shutdown of the administration server instance and indicates that the audit session has been stopped.

## Viewing Security Audit Logs Using BSM Auditing on Solaris

Use the following procedure to enable BSM auditing on Solaris:

1. Login to the Solaris machine as root.
2. Edit the `/etc/security/audit_control` file.
3. Set `naflags` to `lo,ad`.

4. Enable auditing by executing `/etc/security/bsmconv`.
5. Reboot the Solaris machine.

Use the command `praudit` to monitor audit events. Here's an example:

```
tail -f /var/audit/20031219001450.not_terminated.machine1 | praudit
```

You should replace the audit filename (20031219001450.not\_terminated.machine1) with the name of the audit file created on your local machine.

## Quality of Service (QOS)

*Quality of Service* refers to the performance limits you set for a server instance virtual server class or virtual server. For example, if you are an Internet Service Provider (ISP), you might want to charge different fees for virtual servers depending on how much bandwidth is provided. You can limit two areas: the amount of bandwidth and the number of connections.

The Quality of Service information provided by the Sun Java System Application Server is used to determine server efficiency during runtime with respect to:

- Start-up time
- Server traffic and effects of traffic upon bandwidth
- Analysis of live versus static data
- Other data elements

For more information, see [“Administering the Transaction Service Using the CLI” on page 155](#).

## Extracting Monitoring Data Using the CLI

With the `asadmin` command, you can extract monitored data through the command-line interface (CLI) using the `list` and `get` commands.

---

**NOTE** The `set` command is only used to set monitoring for the transaction service as described in [“Administering the Transaction Service Using the CLI” on page 155](#).

---

This section addresses the following topics:

- [list --monitor Command](#)
- [get --monitor Command](#)
- [CLI Name Mapping](#)
- [HTTP Server Monitorable Objects](#)

## list --monitor Command

The `list` command provides information about the application components and subsystems currently being monitored for the specified server instance name. Using this command, you can see the monitorable components and sub-components for a server instance.

### *Example*

```
asadmin> list --monitor server1
```

returns the following list of application components and subsystems that have monitoring enabled:

```
iiop-service  
transaction-service  
application.converter  
application.myApp  
http-server
```

You can also list applications that are currently monitored in the specified server instance. This can be useful when particular monitoring statistics are sought from an application using the `get` command.

### *Example*

```
asadmin> list --monitor server1.application
```

returns:

```
converter  
myApp
```

For a more comprehensive example, see [“Petstore Example”](#) on page 139.



## get --monitor Command

This command retrieves the following monitored information:

- All attribute(s) monitored within a component or subsystem
- Specific attribute monitored within a component or subsystem

When an attribute is requested that does not exist for a particular component or subsystem, an error is returned. Similarly, when a specific attribute is requested that is not active for a component or subsystem, an error is returned.

Refer to [“CLI Name Mapping” on page 138](#) for more information on the use of the `get` command.

### *Example 1*

Attempt to get all attributes from a subsystem for a specific attribute:

```
asadmin> get --monitor server1.iiop-service.orb.system.orb-connection.*
total-inbound-connections=1
total-outbound-connections=1
```

### *Example 2*

Attempt to get all attributes from a J2EE application:

```
asadmin> get --monitor server1.application.converter.*
Attribute name(s) not found
```

There are no monitorable attributes exposed at the J2EE-application level, therefore the command fails.

### *Example 3*

Attempt to get a specific attribute from a subsystem:

```
asadmin> get --monitor server1.transaction-service.inflight-tx
Attribute name = inflight-tx Value = No active transaction found.
```

### *Example 4*

Attempt to get an unknown attribute from within a subsystem attribute:

```
asadmin> get --monitor
server1.iiop-service.orb.system.orb-connection.bad-name
Could not get the attribute
```

```
Execution failed for the command: get --monitor
server1.iiop-service.orb-connection.bad-name
```

## CLI Name Mapping

The Sun Java System Application Server uses a tree structure to track monitorable objects. Every node in the tree has a name and a type. If a type is singleton, only a single node of the type can exist under any parent node. For more information on type of nodes in this tree, see [“Monitorable Object Types” on page 141](#).

The root object in the tree is represented by the Application Server instance name. For example, the root monitoring object for an instance named `server1` is:

```
server1
```

All child objects are then addressed using the dot (.) character as separator. If a child node is of singleton type, then only the monitoring object type is needed to address the object, otherwise a name of the form `type.name` is needed to address the object.

For example, `http-server` is one of the valid monitorable object types and is singleton. To address a singleton child node representing the `http-server` of instance `server1`, the name is:

```
server1.http-server
```

Another example, `application`, is a valid monitorable object type and is not a singleton. To address a non-singleton child node representing the `application` Petstore, the name is:

```
server1.application.petstore
```

The CLI names can also address specific attributes in monitorable objects. For example, `http-server` has a monitorable attribute called `summary`. The following name addresses the `summary` attribute:

```
server1.http-server.summary
```

There is no fixed naming convention for attribute names exposed by monitoring objects.

You are not expected to know the valid names for CLI use. The `list` command lets you inspect available monitorable objects, while the `get` command used with a wildcard parameter allows you to inspect all available attributes on any monitorable object.

The following example illustrate some client name mapping scenarios:

## Petstore Example

A user wants to inspect the number of calls made to a method in the Petstore application deployed on the Sun Java System Application Server instance named `server1`. A combination of the `list` and `get` commands is used to access desired statistics on a method.

1. Invoke the CLI in multi-mode.
2. Set some useful environment variables to avoid entering them for every command:

```
asadmin>export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123
```

```
asadmin>export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=4848
```

3. List monitorable components for instance `server1`:

```
asadmin>list --monitor server1
```

Output is:

```
iiop-service
transaction-service
application.CometEJB
application.ConverterApp
application.petstore
http-server
resources
```

The list of monitorable components includes `iiop-service`, `http-server`, `transaction-service`, `resources`, and all deployed (and enabled) applications.

4. List the monitorable subcomponents in the Petstore application (`-m` can be used instead of `--monitor`):

```
asadmin>list -m server1.application.petstore
```

Output is:

```
ejb-module.signon-ejb_jar
ejb-module.catalog-ejb_jar
ejb-module.uidgen-ejb_jar
ejb-module.customer-ejb_jar
ejb-module.petstore-ejb_jar
ejb-module.AsyncSenderJAR_jar
ejb-module.cart-ejb_jar
```

5. List the monitorable subcomponents in the EJB module `signon-ejb_jar` of the Petstore application:

```
asadmin>list -m
server1.application.petstore.ejb-module.signon-ejb_jar
```

Output is:

```
entity-bean.UserEJB
stateless-session-bean.SignOnEJB
```

6. List the monitorable subcomponents in the entity bean `UserEJB` for the EJB module `signon-ejb_jar` of the Petstore application:

```
asadmin>list -m
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.
UserEJB
```

Output is:

```
bean-method.create0
bean-method.findByPrimaryKey1
bean-method.remove2
bean-method.getUserName3
bean-method.setPassword4
bean-method.getPassword5
bean-method.matchPassword6
bean-method.remove7
bean-method.isIdentical8
bean-method.getEJBLocalHome9
bean-method.getPrimaryKey10
bean-pool
bean-cache
```

7. List the monitorable subcomponents in the method `getUserName3` for the entity bean `UserEJB` in the EJB module `signon-ejb_jar` of the Petstore application:

```
asadmin>list -m
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.
UserEJB.bean-method.getUserName3
```

Output is:

```
No monitorable entities for element
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.
UserEJB.bean-method.getUserName3
```

8. There are no monitorable subcomponents for methods. Get all monitorable statistics for the method `getUserName3`.

```
asadmin>get -m server1.application.petstore.ejb-module.  
signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3.*  
method-name = public abstract java.lang.String  
com.sun.j2ee.blueprints.signon.user.ejb.UserLocal.getUserName()  
total-num-errors = 0  
total-num-success = 2  
execution-time-millis = 1  
total-num-calls = 2
```

9. You can also get a specific statistic, such as execution time.

```
asadmin>get -m server1.application.petstore.ejb-module.  
signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3.executi  
on-time-millis  
execution-time-millis = 1
```

## Monitorable Object Types

The tree of objects used for monitoring contains several nodes. A node is a specific entry in the object tree, and is identified uniquely by its type, name, and parent node. Some of the node types are singleton, meaning that only one node of the type can exist under a parent node. A name is not relevant for a singleton node. A non-singleton type node needs a name.

The following tree describes the tree structure in terms of possible parent-child relationships among various node types and the namespaces for some of the node types.

```

serverInstance
  .application.applicationName
    .ejb-module.moduleName (for x.jar, name is x.jar)
      .stateless-session-bean.beanName
        .bean-pool
        .bean-method.methodName
      .stateful-session-bean.beanName
        .bean-cache
        .bean-method.methodName
      .entity-bean.beanName
        .bean-cache
        .bean-pool
        .bean-method.methodName
      .message-driven-bean.beanName
        .bean-pool
        .bean-method.methodName (methodName = onMessage)
  .http-server
    .process
    .virtual-server
  .iiop-service
    .orb.orbName> (orbName = "system", or userOrbName)
    .orb-connection
    .orb-thread-pool
  .standalone-ejb-module.moduleName
    .stateless-session-bean.beanName
      .bean-pool
      .bean-method.methodName
    .stateful-session-bean.beanName
      .bean-cache
      .bean-method.methodName
    .entity-bean.beanName
      .bean-cache
      .bean-pool
      .bean-method.methodName
    .message-driven-bean.beanName
      .bean-pool
      .bean-method.methodName (methodName = onMessage)
  .transaction-service
  .resources
    .jdbc-connection-pool.poolName

```

## Monitable Attribute Names

It is not necessary for every monitable object to expose monitable attributes. Some of the objects are used just for grouping other objects. For the Sun Java System Application Server, except for an `http-server` node, only the leaf nodes of the tree have attributes. The `http-server` node type has attributes as well as child nodes. The following tables list possible monitable attribute names for various nodes.

**Table 6-1** `http-server`

Attribute Name	Datatype	Description
<code>summary</code>	String (Formatted)	HTTP server summary. Includes virtual servers and processes.  NOTE: See section <a href="#">“HTTP Server Monitable Objects” on page 147</a> for more information on what data is exposed in the formatted string.

**Table 6-2** `virtual-server`

Attribute Name	Datatype	Description
<code>&lt;vs-id&gt;</code>	String (Formatted)	Virtual server information. For every application server instance there can be one or more virtual servers. The list of virtual server IDs can be obtained from the <code>summary</code> attribute of <code>http-server</code> . You can look up statistics for a specific virtual server using the <code>get</code> command parameter of the form <code>server1.http-server.virtual-server.&lt;vs-id&gt;</code> . You can look up statistics for all virtual servers using the <code>get</code> command parameter of the form <code>server1.http-server.virtual-server.*</code> .  NOTE: See section <a href="#">“HTTP Server Monitable Objects” on page 147</a> for more information on what data is exposed in the formatted string.

**Table 6-3** `process`

Attribute Name	Datatype	Description
<code>&lt;pid&gt;</code>	String (Formatted)	Process information. For every application server instance there is one process. The process ID can be obtained from the <code>summary</code> attribute of <code>http-server</code> . Statistics for a process can be obtained using the parameter of the form <code>server1.http-server.process.&lt;pid&gt;</code> to the <code>get</code> command.  NOTE: See section <a href="#">“HTTP Server Monitable Objects” on page 147</a> for more information on what data is exposed in the formatted string.

**Table 6-4** orb-connection

Attribute Name	Datatype	Description
total-inbound-connections	Integer	Total inbound connections to ORB.
total-outbound-connections	Integer	Total outbound connections from ORB.

**Table 6-5** orb-thread-pool

Attribute Name	Datatype	Description
thread-pool-size	Integer	Total number of threads in ORB thread pool.
waiting-thread-count	Integer	Number of thread pool threads waiting for work to arrive.

**Table 6-6** jdbc-connection-pool

Attribute Name	Datatype	Description
total-threads-waiting	Integer	Total threads waiting for JDBC connection.
total-outbound-connections	Integer	Total JDBC connection validation failures.
total-connections-timed-out	Integer	Total connection requests timed out.

**Table 6-7** transaction-service

Attribute Name	Datatype	Description
total-tx-completed	Integer	Total completed transactions.
total-tx-rolled-back	Integer	Total rolled back transactions.
total-tx-inflight	Integer	Total inflight (live) transactions.
isFrozen	String	Is transaction system frozen (true or false)?
inflight-tx	String (Formatted)	List of inflight transactions.



**Table 6-8** bean-pool

Attribute Name	Datatype	Description
max-pool-size	Integer	The maximum number of bean instances in the pool.
steady-pool-size	Integer	The number of bean instances normally maintained in the pool. When a pool is first created, it will be populated with a size equal to steady-pool-size. When an instance is removed from the pool, it is replenished asynchronously, so that the pool size is at, or above, steady-pool-size.
pool-resize-quantity	Integer	The increment by which pool grows up to max-pool-size or shrinks to steady-pool-size
idle-timeout-in-seconds	Integer	Defines the rate at which the pool cleaning thread is executed. Checks if the current size is greater than the steady pool size and removes the pool-resize-quantity elements. If the current size is less than steady-pool-size, it is increased by pool-resize-quantity, with a ceiling of min (current-pool-size + pool-resize-quantity, max-pool-size). Only objects that have not been accessed for more than pool-idle-timeout-in-seconds are candidates for removal
num-beans-in-pool	Integer	Number of beans available in pool.
num-threads-waiting	Integer	Number of threads waiting for free beans.
total-beans-created	Integer	Number of beans created so far.
total-beans-destroyed	Integer	Number of beans destroyed so far.
jms-max-messages-load	Integer	The maximum number of messages to load into a JMS session at one time for a message-driven bean to serve. Default is 1. Applies only to pools for message driven beans.

**Table 6-9** bean-cache

Attribute Name	Datatype	Description
cache-resize-quantity (resize-quantity)	Integer	The quantity by which the cache size is reduced when the number of beans in cache equals max-cache-size (that is, when cache overflow occurs.)
cache-misses	Integer	The number of times a user request did not find a bean in the cache.
idle-timeout-in-seconds	Integer	Rate at which the cache cleaner thread is scheduled. This cleaner thread examines all beans in the cache and passivates those beans that are not accessed for cache-idle-timeout-in-seconds.

**Table 6-9** bean-cache (Continued)

Attribute Name	Datatype	Description
cache-hits	Integer	The number of times a user request found an entry in the cache.
total-beans-in-cache	Integer	The number of beans in the cache. This is the current size of the cache.
max-beans-in-cache	Integer	Maximum number of beans that can be held in the cache beyond which cache overflow occurs.
num-passivations	Integer	Number of passivations. Applies only to stateful session beans.
num-passivation-errors	Integer	Number of errors during passivation. Applies only to stateful session beans.
num-expired-sessions-removed	Integer	Number of expired sessions removed by the cleanup thread. Applies only to stateful session beans.
num-passivation-success	Integer	Number of times passivation completed successfully. Applies only to stateful session beans.

**Table 6-10** bean-method

Attribute Name	Datatype	Description
method-name	String	Fully qualified name of the method.
total-num-calls	Integer	Number of times the method has been invoked. This is collected for stateless and stateful session beans and entity beans if monitoring enabled is true for EJB container, and for message-driven beans if monitoring is enabled for the message-driven bean container.
total-num-errors	Integer	Number of times the method execution resulted in an exception. This is collected for stateless and stateful session beans and entity beans if monitoring is enabled under EJB settings, and for message-driven beans if monitoring is enabled under MDB settings.
total-num-success	Integer	Number of times the method successfully executed. This is collected for stateless and stateful session beans and entity beans if monitoring enabled is true for EJB container, and for message-driven beans if monitoring is enabled for the container.
execution-time-millis	Long	Time spent executing the method for the last successful run of this method. This is collected for stateless and stateful session beans and entity beans if monitoring is enabled on the EJB container and for message-driven beans if monitoring is enabled on the container.

## HTTP Server Monitorable Objects

The HTTP server monitorable attribute name `summary` prints the attribute values of the `Server` element and a summary of its subelements, including the number of each subelement and attribute values for each subelement. The HTTP server `virtual-server` attribute prints the attribute values of the `VirtualServer` element and the details of each of its subelements. The `process` attribute prints the attribute values of the `Process` element and the details of each of its subelements.

To enable NSAPI performance profiling and obtain statistics on the `Profile` and `ProfileBucket` elements, see the *Sun Java System Application Server Developer's Guide to NSAPI*.

For information on how to use the monitoring statistics for performance tuning, see the *Sun Java System Application Server Performance and Tuning Guide*.

### Monitorable HTTP Server Elements

The following table lists the HTTP server monitorable elements.

**Table 6-11** Monitorable HTTP Server Elements

Element Name	Subelements	Description
Server	ConnectionQueue ThreadPool Profile Process VirtualServer	A server instance.
ConnectionQueue	None	The queue in which requests are held prior to being serviced. There is only one connection queue in Sun Java System Application Server.
ThreadPool	None	A thread pool, as defined in the <code>init.conf</code> file.
Profile	None	An NSAPI performance profile bucket, as defined in the <code>init.conf</code> file.
Process	ConnectionQueueBucketThread PoolBucket DnsBucket DnsBucket KeepaliveBucket CacheBucket Thread	A single server process within a server instance.
ConnectionQueueBucket	None	Tracks statistics pertaining to a specific <code>ConnectionQueue</code> .

**Table 6-11** Monitorable HTTP Server Elements (*Continued*)

Element Name	Subelements	Description
ThreadPoolBucket	ThreadPoolBucket	Tracks statistics pertaining to a specific ThreadPool.
DnsBucket	None	Tracks DNS statistics.
KeepaliveBucket	None	Tracks keepalive (persistent connection) statistics.
CacheBucket	None	Tracks file cache (NSFC) statistics.
Thread	RequestBucket ProfileBucket	Describes a request processing thread.
VirtualServer	RequestBucket ProfileBucket	Describes a virtual server.
RequestBucket	None	Tracks request-related statistics.
ProfileBucket		Tracks statistics pertaining to a Profile element.

Monitorable HTTP Server Attributes

The following tables list the HTTP server monitorable attributes.

**Table 6-12** Server

Attribute Name	Values	Description
Id		The server instance ID (for example, server1).
VersionServer		A string that contains the Sun Java System Application Server version.
TimeStarted	GMT	The time this server instance was started.
SecondsRunning		The number of seconds since this server instance started.
TicksPerSecond		The number of ticks in a second. This value is system-dependent.
MaxProcs		The maximum number of processes.
MaxThreads		The maximum number of processing threads.
MaxVirtualServers		The maximum number of virtual servers tracked.
FlagProfilingEnabled	0 (off), 1 (on)	Indicates whether NSAPI performance profiling is enabled (on).

**Table 6-12** Server (Continued)

Attribute Name	Values	Description
FlagVirtualServerOverflow	0 (no), 1 (yes)	Indicates whether more than <code>MaxVirtualServers</code> are configured (yes). If this attribute is set to 1, statistics are not being tracked for all virtual servers.
LoadMinuteAverage		Average load in 1 minute.
Load5MinuteAverage		Average load in 5 minutes.
Load15MinuteAverage		Average load in 15 minutes.
RateBytesTransmitted	bytes per second	The rate at which data is transmitted over some server-defined interval, or 0 if this information is not available.
RateBytesReceived	bytes per second	The rate at which data is received over some server-defined interval, or 0 if this information is not available.

**Table 6-13** ConnectionQueue

Attribute Name	Values	Description
Id		The connection queue ID.

**Table 6-14** ThreadPool

Attribute Name	Values	Description
Id		The thread pool ID.
Name		The symbolic name of the thread pool.

**Table 6-15** Profile

Attribute Name	Values	Description
Id		The NSAPI performance profile bucket ID.
Name		The symbolic name of the NSAPI performance profile bucket.
Description		The description of the NSAPI performance profile bucket.

**Table 6-16** Process

Attribute Name	Values	Description
Pid		The operating system process identifier that uniquely identifies this process.
Mode	unknown active	Displays active when this process is active.
TimeStarted	GMT	The time this process was started.
CountConfigurations		The number of times a configuration has been loaded, or 0 if this information is not available.
SizeVirtual	kilobytes	The size of virtual memory used by this process.
SizeResident	kilobytes	The size of the resident memory used by this process.
FractionSystemMemoryUsage		Fraction of system memory used by this process.

**Table 6-17** ConnectionQueueBucket

Attribute Name	Values	Description
ConnectionQueue		The ID of a <code>ConnectionQueue</code> element.
CountTotalConnection		The total number of new connections that have been accepted.
CountQueued		The number of connections currently enqueued.
PeakQueued		The largest number of connections that have been in the queue simultaneously.
MaxQueued		The maximum number of connections that can be in the queue.
CountOverflow		The number of times the queue has been too full to accommodate a connection.
CountTotalQueued		The total number of connections that have been queued. A given connection may be queued multiple times, so <code>CountTotalQueued</code> may be greater than or equal to <code>CountTotalConnections</code> .
TicksTotalQueued		The total number of ticks that connections have spent in the queue. A tick is a system-dependent unit of time; see <code>TicksPerSecond</code> .

**Table 6-18** ThreadPoolBucket

Attribute Name	Values	Description
Thread-pool		The ID of a <code>ThreadPool</code> element.
CountThreadsIdle		The number of request processing threads currently idle.
CountThreads		The number of request processing threads.
MaxThreads		The maximum number of request processing threads that can exist concurrently.
CountQueued		The number of requests queued for processing by this thread pool.
PeakQueued		The largest number of requests that have been in the queue simultaneously.
MaxQueued		The maximum number of requests that can be in the queue.

**Table 6-19** DnsBucket

Attribute Name	Values	Description
FlagCacheEnabled	0 (off), 1 (on)	Indicates whether the DNS cache is enabled (on).
CountCacheEntries		The number of DNS entries presently in the cache.
MaxCacheEntries		The maximum number of DNS entries the cache can accommodate.
CountCacheHits		The number of times a DNS cache lookup has succeeded.
CountCacheMisses		The number of times a DNS cache lookup has failed.
FlagAsyncEnabled	0 (off), 1 (on)	Indicates whether asynchronous DNS lookups are enabled (on).
CountAsyncNameLookups		The total number of asynchronous DNS name lookups performed.
CountAsyncAddrLookups		The total number of asynchronous DNS address lookups performed.
CountAsyncLookupsInProgress		The total number of asynchronous DNS lookups currently in progress.

**Table 6-20**   KeepaliveBucket

Attribute Name	Values	Description
CountConnections		The number of connections currently in keepalive mode.
MaxConnections		The maximum number of simultaneous keepalive connections.
CountHits		The total number of times connections in keepalive mode have subsequently made a valid request.
CountFlushes		The number of times keepalive connections have been closed by the server.
CountTimeouts		The number of times keepalive connections have timed out.
SecondsTimeouts		The number of seconds before the server closes an idle keepalive connection.
CountRefusals		The number of times keepalive connections have been refused by the server.

**Table 6-21**   CacheBucket

Attribute Name	Values	Description
FlagEnabled	0 (off), 1 (on)	Indicates whether the file cache is enabled (on).
SecondsMaxAge	Number of seconds	The maximum age of a file cache entry.
CountEntries		The number of entries currently in the file cache.
MaxEntries		The maximum number of cache entries the file cache can accommodate simultaneously.
CountOpenEntries		The number of entries associated with an open file.
MaxOpenEntries		The maximum number of cache entries associated with an open file that the file cache can accommodate simultaneously.
SizeHeapCache	Number of bytes	The amount of heap used by cached file content.
MaxHeapCacheSize	Number of bytes	The maximum amount of heap the file cache uses for cached file content.
SizeMmapCache	Number of bytes	The amount of address space used by memory mapped file content.



**Table 6-21** CacheBucket (*Continued*)

Attribute Name	Values	Description
MaxMmapCacheSize	Number of bytes	The maximum amount of address space that the file cache uses for memory mapped file content.
CountHits		The number of times a cache entry lookup has succeeded.
CountMisses		The number of times a cache entry lookup has failed.
CountInfoHits		The number of times a file information lookup has succeeded.
CountInfoMisses		The number of times a file information lookup has failed.
CountContentHits		The number of times a content lookup has succeeded.
CountContentMisses		The number of times a content lookup has failed.

**Table 6-22** Thread

Attribute Name	Values	Description
Mode	unknown, idle, DNS, request, processing, response, updating	The thread's last known status.
TimeStarted	GMT	The time this thread was started.
ConnectionQueue		The ID of the <code>ConnectionQueue</code> the thread is servicing.

**Table 6-23** VirtualServer

Attribute Name	Values	Description
Id		The virtual server ID.
Mode	unknown, active	Displays active when this virtual server is active.
Hosts		The software virtual server hostnames serviced by this virtual server (for example, www.foo.com foo.com foo.isp.com).
Interfaces		The interfaces (listeners) the virtual server is configured for (for example, 192.168.1.2:80 192.168.1.2:443).

**Table 6-24** RequestBucket

Attribute Name	Values	Description
CountRequests		The number of requests serviced.
CountBytesReceived		The number of bytes received, or 0 if this information is not available.
CountBytesTransmitted		The number of bytes transmitted, or 0 if this information is not available.
RateBytesTransmitted	bytes per second	The rate at which data was transmitted over some server-defined interval, or 0 if this information is not available.
MaxByteTransmissionRate		The maximum rate at which data was transmitted over some server-defined interval, or 0 if this information is not available.
CountOpenConnections		The number of open connections, or 0 if this information is not available.
MaxOpenConnections		The maximum number of open connections, or 0 if this information is not available.
Count2xx		The number of 200-level responses sent.
Count3xx		The number of 300-level responses sent.
Count4xx		The number of 400-level responses sent.
Count5xx		The number of 500-level responses sent.
CountOther		The number of responses sent that were not 200, 300, 400, or 500 level.
Count200		The number of 200-level responses sent.
Count302		The number of 302-level responses sent.
Count304		The number of 304-level responses sent.
Count400		The number of 400-level responses sent.
Count401		The number of 401-level responses sent.
Count403		The number of 403-level responses sent.
Count404		The number of 404-level responses sent.
Count503		The number of 503-level responses sent.

**Table 6-25** ProfileBucket

Attribute Name	Values	Description
Profile		The ID of a <code>Profile</code> element.
Countcalls		The number of calls to NSAPI SAFs.
CountRequests		The number of requests processed.
TicksDispatch		The number of ticks spent dispatching requests. A tick is a system-dependent unit of time; see <code>TicksPerSecond</code> .
TicksFunction		The number of ticks spent in NSAPI SAFs. A tick is a system-dependent unit of time; see <code>TicksPerSecond</code> .

## Administering the Transaction Service Using the CLI

You can use the `set` command to administer the statistics you want to monitor for the JTS.

### *Example 1*

To add a transaction to a rollback list (which results in rollback or specified transaction), issue the `set` command as follows:

```
set --monitor server1.transaction-service.rollback-list=txnidl
```

### *Example 2*

To freeze the transaction service, issue the `set` command as follows:

```
set --monitor server1.transaction-service.freeze=true
```

The attributes that you can monitor can be set from the command line according to the rules described in [“CLI Name Mapping” on page 138](#).

For more information on the java transaction service, refer to [Chapter 8, “Using Transaction Services.”](#)

# Using HTTP Quality of Service

The following settings govern how traffic is counted and how often the bandwidth is recomputed:

- **Recompute interval**—Indicates how often (in milliseconds) the bandwidth is computed.
- **Metric interval**—The period of time for which data is used in traffic calculations.

In the Administration interface, you can enable these server or class-level settings for the server instance or for a class of virtual servers. However, you can override them for an individual virtual server.

This section includes the following topics:

- [Quality of Service Example](#)
- [Configuring Quality of Service \(QOS\)](#)
- [Required Changes to the obj.conf File](#)
- [Known Limitations to Quality of Service](#)

## Quality of Service Example

The following example shows how the Quality of Service information is collected and computed.

- The server has metric interval of 30 seconds.
- The server starts up at a time of 0 seconds.
- At 1 second, an HTTP connection generates 5000 bytes of traffic to/from the server.
- No more connections are made after that. At 30 seconds, the total traffic for the last 30 seconds is 5000 bytes.
- At 32 seconds, the traffic sample from 1 second is discarded, since it is older than the 30 seconds of the metric interval. The total traffic for the last 30 seconds is now 0.

The recompute interval works similarly. The server's recompute interval is 100ms.

Continuing with the example, the bandwidth is recomputed periodically every 100 milliseconds. The calculation is based on the amount of traffic as well as the metric interval.

- At 0 seconds, the bandwidth is calculated for the first time. The total traffic is zero, divided by the metric interval of 30 seconds, giving a bandwidth of zero.
- At 1 second, the bandwidth is calculated for the 10th time (1000 milliseconds/100 milliseconds). The total traffic is 5000 bytes, which is divided by 30 seconds. The bandwidth is  $5000/30 = 166$  bytes per second.
- At 30 seconds, the bandwidth is calculated for the 300th time. The total traffic is 5000 bytes, which is divided by 30 seconds. The bandwidth is  $5000/30 = 166$  bytes per second.
- At 32 seconds, the bandwidth is computed again for the 320th time. The traffic is now 0 (since the one connection that generated traffic is too old to be counted), divided by 30, giving a bandwidth of 0 bytes/second.

## Configuring Quality of Service (QOS)

Quality of Service for a server instance or a class of virtual servers is configured through the Administration interface.

---

**NOTE** To enforce your Quality of Service settings, you must also set up Server Application Functions (SAFs) in the `obj.conf` file as described in [“Required Changes to the obj.conf File” on page 159](#).

---

To configure Quality of Service, follow these steps:

1. Select the App Server Instances node in the left pane.
2. Expand the server instance node to display the HTTP Server node.
3. Click the HTTP Server node to display the QOS tab.
4. Click the QOS tab.

A page displays the general settings for Quality of Service, followed by a properties button.

5. To enable Quality of Service for this HTTP Server, click QOS Enable.

Note: By default, Quality of Service is disabled. Enabling Quality of Service slightly increases server overhead.

**6. Specify the QOS Metrics Interval.**

The metrics interval is the time period, in seconds, that data is sampled during server traffic calculations. The default value is 30 seconds.

If your site commonly has large file transfers, use a large value (several minutes or more) for this field. A large file transfer might take up all the allowed bandwidth for a short metric interval, and result in connections being denied if you've enforced the maximum bandwidth setting. Since the bandwidth is averaged by the metric interval, a longer interval smooths out spikes caused by large files.

If the bandwidth limit is much lower than available bandwidth (for example, 1 MB-per-second bandwidth limit but with a 1 GB-per-second connection to the backbone), the metric interval should be shortened.

Note: If you have large static file transfers and a bandwidth limit that is much lower than the available bandwidth, you must decide which situation to tune for, since the problems require opposite solutions.

**7. Specify the QOS Recompute Time Interval.**

The recompute time interval is the number of milliseconds between each computation of the bandwidth for all servers, classes, and virtual servers. The default is 100 milliseconds.

**8. Specify the Bandwidth Limit.**

This is the maximum bandwidth limit for the server instance in bytes per second. It is interdependent with the QOS Metrics Interval to some extent.

**9. Choose whether or not to enforce the maximum bandwidth setting.**

If you choose to enforce the maximum bandwidth, when the server reaches its bandwidth limit, additional connections are refused.

If you do not enforce the maximum bandwidth, when the maximum is exceeded, the server logs a message to the event log.

**10. Specify the connection limit.**

This is the number of concurrent requests processed.

**11. Choose whether or not to enforce the connection limit setting.**

If you choose to enforce the maximum connections, when the server reaches its limit, additional connections are refused. If you do not enforce the maximum connections, when the maximum is exceeded, the server logs a message to the event log.

12. (Optional). If specifying additional name-value pair attributes for Quality of Service, click the Properties button.

For a list of allowed name-value pairs for Quality of Service properties, refer to the online help.

13. Click Save to commit the changes to the server instance.
14. Access App Server Instances and your server instance in the left pane, then click Apply Changes.

## Required Changes to the obj.conf File

To enforce Quality of Service, you must include directives in your `obj.conf` file to invoke following Server Application Functions (SAFs):

- `AuthTrans qos-handler`
- `Error qos-error`

In order to work properly, the `qos-handler` `AuthTrans` directive must be the first `AuthTrans` configured in the default object. The role of the Quality of Service handler is to examine the current statistics for the virtual server, virtual server class, and global server, and to enforce the limits by returning an error. The Sun Java System Application Server includes a built-in sample Quality of Service handler SAF, called `qos-handler`. This SAF logs when limits are reached, and returns a 503 `Server busy` error to the server so that it can be processed by the NSAPI.

The Sun Java System Application Server also includes a built-in sample error SAF called `qos-error` which returns an error page stating which limits caused the 503 error and the value of the statistic that triggered the limit.

For more information on these SAFs and how to use them, see the *Sun Java System Application Server Developer's Guide to NSAPI*.

## Known Limitations to Quality of Service

When you use the Quality of Service features, keep in mind the following limitations:

- The Quality of Service features only measure the HTTP bandwidth at the application level. The HTTP bandwidth can differ from the actual TCP network bandwidth for a variety of reasons:

- If SSL is enabled, handshakes and client certificate exchanges add to the traffic but are not measured.
- If chunked encoding is enabled in either or both directions, the chunking layer removes the chunk headers and they are not counted in the traffic. Other headers or protocol items are counted.
- The Quality of Service features cannot accurately measure traffic from `PR_TransmitFile` calls. For basic I/O operations such as `PR_Send()/net_write` or `PR_Recv()/net_read`, the data transferred can be quickly accounted for by the bandwidth manager, since the number of bytes transferred in one system call is usually the size of a buffer and the I/O call returns quickly. This works very well to measure the instantaneous bandwidth of dynamic content applications. However, because the amount of data transferred from `PR_TransmitFile` is only known at the end of the transfer, it can't be measured before it completes.

If the `PR_TransmitFile` is short, then the Quality of Service features will perform adequately. However, If the `PR_TransmitFile` is long, as in the case of a long file downloaded by a dialup user, the whole amount of data transferred will be counted at completion time. When the bandwidth manager recomputes bandwidth after the next recompute interval period starts, the bandwidth computed will increase significantly because of that recent large `PR_TransmitFile`. This case could cause the server to deny all requests until the next metric interval, when the bandwidth manager will "expire" the transmit file operation (since it is too old) and thus the bandwidth value will go back down. If your site commonly has very long static file downloads, you should increase the metric interval from the default 30 seconds.

- The bandwidth computed is always an approximation because it is not measured instantaneously, but is recomputed at regular intervals and over a certain period. For example, if the metric interval is the default 30 seconds and the server is idle for 29 seconds, then the next second, a client could potentially use 30 times the bandwidth limit in one second.
- The Quality of Service bandwidth statistics are lost whenever the server is reconfigured dynamically. In addition, the Quality of Service limitations are not enforced in threads that have connections on an older, inactive configuration, because the bandwidth manager thread only computes bandwidth statistics for the active configuration. Potentially, a client that doesn't close its socket for a long time and remains active so that the server doesn't time it out would not be subject to the Quality of Service limitations after a dynamic server reconfiguration.



- The concurrent connections are computed with a different granularity for virtual servers than for virtual server classes and the global server instance. The connection counter for an individual virtual server is incremented atomically immediately after the request is parsed and routed to the virtual server. It is also decremented atomically at the end of the response processing for that request. This means that the virtual server connection statistics are always exact at any instant.

However, the connection statistics for the virtual server class and global server instance are not updated instantly. They are updated by the bandwidth manager thread every recompute interval. The connection count for the virtual server class is the sum of the connections on all virtual servers of that class; and the global server instance connection count is the sum of connections on all virtual server classes.

Because of the way these values are computed, the number of connections for a virtual server is always correct, and if you've enforced a limit to the number of connections, you can never have more than the limit. The virtual server class and server instance values are not quite as accurate, since they are only computed at intervals.

## About SNMP

Simple Network Management Protocol (SNMP) is a protocol used to exchange management and monitoring information across a network. With SNMP, data travels between a managed device and a network management station (NMS). A *managed device* is anything that runs SNMP: hosts, routers, your HTTP server, and other servers on your network.

This section addresses the following topics:

- [Network Management Station \(NMS\)](#)
- [Management Information Base \(MIB\) Objects](#)
- [SNMP Messages](#)
- [SNMP Trap Destinations](#)
- [SNMP Agent Community](#)

## Network Management Station (NMS)

The *network management station* (NMS) is a machine used to remotely manage a specific network. Usually, the NMS software will provide a graph to display collected data or use that data to make sure the server is operating within a particular tolerance.

The NMS is usually a powerful workstation with one or more network management applications installed. A network management application such as HP OpenView graphically shows information about managed devices, such as your HTTP servers. For example, it might show which servers in your enterprise are up or down, or the number and type of error messages received. When you use SNMP with the Sun Java System Application Server, this information is transferred between the NMS and the server through the use of two types of agents: the subagent and the master agent.

The subagent gathers information about the server instances running in various domains and passes the information to the master agent. There is a master agent and a subagent for every installation of the Sun Java System Application Server.

---

<b>NOTE</b>	After making any SNMP configuration changes, you must click the Apply button, then restart the SNMP subagent.
-------------	---

---

The *master agent* exchanges information between the various subagents and the NMS. The master agent is installed with the Sun Java System Application Server.

You can have multiple subagents installed on a host computer, but only one master agent. For example, if you had Sun ONE Directory Server, Sun Java System Application Server, and the Sun ONE Messaging Server installed on the same host, the subagents for each of the servers would communicate with the same master agent.

The NMS either requests information from the server or changes the value of a variable store in the server's MIB. For example:

1. The NMS sends a message to the Admin Server master agent. The message might be a request for data (a `GET` message), or an instruction to set a variable in the MIB (a `SET` message).
2. The master agent forwards the message to the appropriate subagent.
3. The subagent retrieves the data or changes the variable in the MIB.
4. The subagent reports data or status to the master agent, then the master agent forwards the message back (a `GET` message) to the NMS.

5. The NMS displays the data textually or graphically through its network management application.

## Management Information Base (MIB) Objects

The Sun Java System Application Server stores variables pertaining to managing and monitoring information across a network. Variables the master agent can access are called managed objects. These objects are defined in a tree-like structure called the *management information base* (MIB). The MIB provides access to the HTTP server's network configuration, status, and statistics. Using SNMP, you can view this information from the network management workstation (NMS).

The top level of the MIB tree shows that the internet object identifier has the following subtrees:

- directory (1)
- mgmt (2)
- experimental (3)
- private (4)

The private (4) subtree contains the enterprises (1) node. Each subtree in the enterprises (1) node is assigned to an individual enterprise, which is an organization that has registered its own specific MIB extensions. An enterprise can then create product-specific subtrees under its subtree. MIBs created by companies are located under the enterprises (1) node.

Each Sun Java System Application Server subagent provides a MIB for use in SNMP communication. The server reports significant events to the NMS by sending messages or traps containing these variables. The NMS can query the server's MIB for data.

Each Sun Java System Application Server has its own MIB located at *install\_dir/lib*

The Sun Java System Application Server's MIB is a file called `appserv.mib`. This MIB contains the definitions for various variables pertaining to network management for the Sun Java System Application Server.

The Sun Java System Application Server MIB has an object identifier of

`appserver 1 (as appserver7 OBJECT IDENTIFIER ::= {appserver 1 })` and is located in the *install\_dir/lib* directory.

You can see administrative information about your Sun Java System Application Server and monitor the server in real time using the Sun Java System Application Server MIB. The following table lists and describes the managed objects stored in the `appserv.mib` file.

**Table 6-26** `appserv.mib` Managed Objects and Descriptions

Managed object	Description
<code>iwsCpuID</code>	CPU identifier.
<code>iwsCpuIdleTime</code>	Idle CPU time.
<code>iwsCpuKernelTime</code>	CPU kernel time.
<code>iwsCpuTable</code>	Application Server CPUs.
<code>iwsCpuUserTime</code>	CPU user time.
<code>iwsInstanceTable</code>	Application Server instances.
<code>iwsInstanceId</code>	Server instance identifier
<code>iwsInstanceVersion</code>	String, such as <code>SunONE-ApplicationServer-Enterprise/7 BB1-01/24/2001 17:15 (SunOS DOMESTIC)</code>
<code>iwsInstanceDescription</code>	Description of the server instance.
<code>iwsInstanceOrganization</code>	Organization responsible for the server instance.
<code>iwsInstanceContact</code>	Contact information for person(s) responsible for server instance.
<code>iwsInstanceLocation</code>	Where the server is located.
<code>iwsInstanceStatus</code>	Status of the server instance.
<code>iwsInstanceUptime</code>	How long the server has been running.
<code>iwsInstanceDeathCount</code>	Number of times server instance processes have gone down.
<code>iwsInstanceRequests</code>	Number of requests processed by the server instance.
<code>iwsInstanceInOctets</code>	Number of octets received by the server instance. Will show 0 if information is not available.
<code>iwsInstanceOutOctets</code>	Number of octets transmitted by the server instance. Will show 0 if information is not available.
<code>iwsInstanceCount2xx</code>	Number of 200-level (Successful) responses issued by the server instance.
<code>iwsInstanceCount3xx</code>	Number of 300-level (Redirection) responses issued by the server instance.
<code>iwsInstanceCount4xx</code>	Number of 400-level (Client Error) responses issued by the server instance.

**Table 6-26** appserv.mib Managed Objects and Descriptions (*Continued*)

Managed object	Description
iwsInstanceCount5xx	Number of 500-level (Server Error) responses issued by the server instance.
iwsInstanceCountOther	Number of other (neither 2xx, 3xx, 4xx, nor 5xx) responses issued by the server instance.
iwsInstanceCount200	Number of 200 (OK) responses issued by the server instance.
iwsInstanceCount302	Number of 302 (Moved Temporarily) responses issued by the server instance.
iwsInstanceCount304	Number of 304 (Not Modified) responses issued by the server instance.
iwsInstanceCount400	Number of 400 (Bad Request) responses issued by the server instance.
iwsInstanceCount401	Number of 401 (Unauthorized) responses issued by the server instance.
iwsInstanceCount403	Number of 403 (Forbidden) responses issued by the server instance.
iwsInstanceCount404	Number of 404 (Not Found) responses issued by the server instance.
iwsInstanceLoad1MinuteAverage	1 minute load average of the system running the server instance.
iwsInstanceLoad5MinuteAverage	5 minute load average of the system running the server instance.
iwsInstanceLoad15MinuteAverage	15 minute load average of the system running the server instance.
iwsInstanceNetworkInOctets	Number of octets transmitted on the network per second.
iwsInstanceNetworkOutOctets	Number of octets received on the network per second.
iwsVsTable	Server virtual servers.
iwsVsId	Virtual server identifier.
iwsVsRequests	Number of requests processed by the virtual server.
iwsVsInOctets	Number of octets received by the virtual server.
iwsVsOutOctets	Number of octets transmitted by the virtual server.
iwsVsCount2xx	Number of 200-level (Successful) responses issued by the virtual server.
iwsVsCount3xx	Number of 300-level (Redirection) responses issued by the virtual server.

**Table 6-26** appserv.mib Managed Objects and Descriptions (*Continued*)

Managed object	Description
iwsVsCount4xx	Number of 400-level (Client Error) responses issued by the virtual server.
iwsVsCount5xx	Number of 500-level (Server Error) responses issued by the virtual server.
iwsVsCountOther	Number of other (neither 2xx, 3xx, 4xx, nor 5xx) responses issued by the virtual server.
iwsVsCount200	Number of 200 (OK) responses issued by the virtual server.
iwsVsCount302	Number of 302 (Moved Temporarily) responses issued by the virtual server.
iwsVsCount304	Number of 304 (Not Modified) responses issued by the virtual server.
iwsVsCount400	Number of 400 (Bad Request) responses issued by the virtual server.
iwsVsCount401	Number of 401 (Unauthorized) responses issued by the virtual server.
iwsVsCount403	Number of 403 (Forbidden) responses issued by the virtual server.
iwsVsCount404	Number of 404 (Not Found) responses issued by the virtual server.
iwsProcessTable	Application Server processes.
iwsProcessId	Operating system process identifier.
iwsProcessThreadCount	Number of request processing threads.
iwsProcessThreadIdle	Number of request processing threads currently idle.
iwsProcessConnectionQueueCount	Number of connections currently in connection queue.
iwsProcessConnectionQueuePeak	Largest number of connections that have been queued simultaneously.
iwsProcessConnectionQueueMax	Maximum number of connections allowed in connection queue.
iwsProcessConnectionQueueTotal	Number of connections that have been accepted.
iwsProcessConnectionQueueOverflows	Number of connections rejected due to connection queue overflow.
iwsProcessKeepaliveCount	Number of connections currently in keepalive queue.
iwsProcessKeepaliveMax	Maximum number of connections allowed in keepalive queue.
iwsProcessSizeVirtual	Process size in kilobytes.

**Table 6-26** appserv.mib Managed Objects and Descriptions (*Continued*)

Managed object	Description
iwsProcessSizeResident	Process resident size in kilobytes.
iwsProcessFractionSystemMemoryUsage	Fraction of process memory in system memory.
iwsListenTable	Application Server listen sockets.
iwsListenId	Listen socket identifier.
iwsListenAddress	Address where socket listens.
iwsListenPort	Port where socket listens.
iwsListenSecurity	Encryption support.
iwsThreadPoolCount	Number of requests denied.
iwsThreadPoolMax	Maximum number of requests allowed in queue.
iwsThreadPoolPeak	Largest number of requests that have been queued simultaneously.
iwsThreadPoolTable	Application Server thread pools.
iwsVsCount503	Number of 503 (Unavailable) responses issued.
iwsInstanceCount503	Number of 503 (Unavailable) responses issued.

## SNMP Messages

**GET** and **SET** are two types of messages defined by SNMP.

Each object is assigned a unique identifier within the MIB. Objects are accessed by the SNMP Manager by issuing **GET** and **GETNEXT** commands which specify the object's unique identifier. The Proxy Agent obtains the value of the specified object and transmits it to the SNMP manager. Events added to the log may generate SNMP traps provided they satisfy trap filter conditions. Events that do not generate traps are merely recorded as an entry in the maintenance log table and are accessed by the SNMP Manager through normal **GET** and **GETNEXT** commands.

**GET** and **SET** messages are sent by a network management station (NMS) to a master agent. You can use one or the other, or both, through the Administration interface.

SNMP exchanges network information in the form of protocol data units (PDUs). These units contain information about variables stored on the managed device, such as the HTTP server. These variables, also known as managed objects, have values and titles that are reported to the NMS as necessary. Protocol data units sent by the server to the NMS are known as *traps*. The use of **GET**, **SET**, and trap messages are explained further in the following sections.

## SNMP Trap Destinations

An *SNMP trap* is a message the SNMP agent sends to a network management station (NMS). For example, an SNMP agent sends a trap when the status of an interface has changed from up to down. The SNMP agent must know the address of the NMS so it knows where to send traps.

You can configure this trap destination for the SNMP master agent from the Sun Java System Application Server Administration interface. You can also view, edit, and remove the trap destinations you have already configured. When you configure trap destinations using the Administration interface, you are actually editing the `CONFIG` file.

The server subagent sends a message or trap to the NMS when a significant event has occurred. For example:

1. The subagent informs the master agent that the server has stopped.
2. The master agent sends a message, or trap, reporting the event to the NMS.
3. The NMS displays the information textually or graphically through its network management application.

Refer to [“Installing the SNMP Master Agent” on page 172](#) for instructions on setting up the SNMP trap port.

## SNMP Agent Community

The *SNMP agent community* consists of a community string and the operations assigned to the specified community. The community string is a text string for a network management station (NMS) name that an SNMP agent uses for authorization. This means that an NMS would send a community string with each message it sends to the agent.

The operations assigned are `get` and/or `set`. The SNMP agent can then verify whether the NMS is authorized to perform `get`, `set`, or both `get` and `set` operations for data exchange. Community strings are not concealed when sent in SNMP packets; strings are sent in ASCII text.

You can configure and manage the community string and the allowed operations for each specified community from the Administration interface. Refer to [“Installing the SNMP Master Agent” on page 172](#) for instructions on setting up the SNMP Agent Community.



# Setting Up SNMP

In general, to use SNMP, a master agent and at least one subagent must be installed and running on your system. The master agent must be installed before you can enable a subagent. Refer to [“Installing the SNMP Master Agent” on page 172](#).

The procedures for setting up SNMP are different depending upon your system. The following table provides an overview of procedures to follow for different situations. The actual procedures are described in detail later in the chapter.

**Table 6-27** Procedure for Setting Up SNMP

If your server meets these conditions...	...follow these procedures. (discussed in detail in the following sections).
No native agent is currently running	<ol style="list-style-type: none"> <li>1. Start the master agent.</li> <li>2. Enable the subagent for each server installed on the system.</li> </ol>
<ul style="list-style-type: none"> <li>• Native agent is currently running</li> <li>• No SMUX</li> <li>• No need to continue using native agent</li> </ul>	<ol style="list-style-type: none"> <li>1. Stop the native agent when you install the master agent for your Administration Server.</li> <li>2. Start the master agent.</li> <li>3. Configure the SNMP subagent for each server instance.</li> </ol>
<ul style="list-style-type: none"> <li>• Native agent is currently running</li> <li>• No SMUX</li> <li>• Needs to continue using native agent</li> </ul>	<ol style="list-style-type: none"> <li>1. Install a proxy SNMP agent.</li> <li>2. Start the proxy SNMP agent.</li> <li>3. Restart the native agent using a port number other than the master agent port number.</li> <li>4. Start the master agent.</li> <li>5. Enable the subagent for each server installed on the system.</li> </ol>

Before you begin, you should verify two things:

- Is your system already running an SNMP agent (an agent native to your operating system)?
- If so, does your native SNMP agent support SMUX communication?

See your system documentation for information on how to verify this information.

- 
- |             |  |
|-------------|--|
| <b>NOTE</b> | After changing SNMP settings in the Admin Server, installing a new server, or deleting an existing server, you must perform the following steps:   |
|             | <ul style="list-style-type: none"><li>• (Windows 2000) Restart the Windows SNMP service or reboot the machine.</li><li>• (UNIX) Restart the SNMP master agent and the SNMP subagent using the Administration Server.</li></ul> |
- 

The following topics are addressed in this section:

- [Using a Proxy SNMP Agent](#)
- [Installing the SNMP Master Agent](#)

## Using a Proxy SNMP Agent

You need to use a proxy SNMP agent when you already have a native agent running, and you want to continue using it concurrently with a Sun Java System Application Server master agent. Before you start, be sure to stop the native master agent. (See your system documentation for detailed information.)

- 
- |             |   |
|-------------|---|
| <b>NOTE</b> | To use a proxy agent, you'll need to install it and then start it. You'll also have to restart the native SNMP master agent using a port other than the port the Sun Java System Application Server master agent is running on. |
|-------------|---|
- 

This section includes the following topics:

- [Installing the Proxy SNMP Agent](#)
- [Starting the Proxy SNMP Agent](#)
- [Restarting the Native SNMP Daemon](#)

### Installing the Proxy SNMP Agent

If an SNMP agent is running on your system and you want to continue using the native SNMP daemon, follow the steps in these sections:

1. Install the SNMP master agent. See [“Installing the SNMP Master Agent” on page 172.](#)
2. Install and start the proxy SNMP agent and restart the native SNMP daemon. See [“Using a Proxy SNMP Agent” on page 170.](#)
3. Start the SNMP master agent. See [“Enabling and Starting the SNMP Master Agent” on page 173.](#)
4. Enable the subagent. See [“Enabling the Subagent” on page 178.](#)

To install the SNMP proxy agent, edit the `CONFIG` file (you can give this file a different name), located in `install_dir/lib/snmp/sagt` in the server root directory, so that it includes the listening port for the SNMP daemon. The proxy agent also needs to include the MIB trees and traps that the proxy SNMP agent will forward.

Here is an example of a `CONFIG` file:

```
AGENT AT PORT 1161 WITH COMMUNITY public
SUBTREES 1.3.6.1.2.1.1,
          1.3.6.1.2.1.2,
          1.3.6.1.2.1.3,
          1.3.6.1.2.1.4,
          1.3.6.1.2.1.5,
          1.3.6.1.2.1.6,
          1.3.6.1.2.1.7,
          1.3.6.1.2.1.8
FORWARD ALL TRAPS;
```

## Starting the Proxy SNMP Agent

To start the proxy SNMP agent, enter the following at the command prompt:

```
# sagt -c CONFIG&
```

## Restarting the Native SNMP Daemon

After starting the proxy SNMP agent, restart the native SNMP daemon at the port you specified in the `CONFIG` file.

To restart the native SNMP daemon, enter the following at the command prompt:

```
# snmpd -P port_number
```

where *port\_number* is the port number specified in the `CONFIG` file. For example, on the Solaris platform, using the port in the previously mentioned example of a `CONFIG` file, you would enter:

```
# snmpd -P 1161
```

## Installing the SNMP Master Agent

---

**NOTE** You cannot use the Administration interface to install and start the master SNMP agent unless the server is running as `root`.

---

To install the master SNMP agent:

1. Log in as `root`.
2. Check whether an SNMP daemon (`snmpd`) is running on port 161.  
If no SNMP daemon is running, go to Step 4.  
If an SNMP daemon is running, make sure you know how to restart it and which MIB trees it supports.
3. If an SNMP daemon is running, kill its process.
4. In the Administration interface, select the Admin Server node from the left pane.
5. Select the Monitoring tab to display the SNMP Agent Trap page.  
The Manager Entries information is displayed on this page.
6. Type the name of the system that is running your network management software.
7. Type the trap port number at which your network management system listens for traps. (The well-known port is 162.) For more information on traps, see [“SNMP Trap Destinations” on page 168](#).
8. Type the community string you want to use in the trap. For more information on community strings, see [“SNMP Agent Community” on page 168](#).
9. Click OK.
10. Click the SNMP Agent Community link within the Monitoring tab.  
The Community Strings information is displayed.

11. Type the community string for the master agent.
12. Choose an operation level for the community.  
After you establish a community, you can edit its settings or remove it from the buttons indicated within the Current Communities heading on this page.
13. Click OK.
14. Access App Server Instances and your server instance in the left pane, then click Apply Changes.

## Enabling and Starting the SNMP Master Agent

Master agent operation is defined in an agent configuration file named `CONFIG`, which you can edit manually. You must install the master SNMP agent before you can enable the SNMP subagent.

---

<b>NOTE</b>	If you get a bind error similar to <code>System Error: Could not bind to port</code> when restarting the master agent, use <code>ps -ef   grep snmp</code> to check if <code>magt</code> is running. If it is running, use the command <code>kill -9 pid</code> to end the process. The CGIs for SNMP will then start working again.
-------------	--

---

This section includes the following topics:

- [Starting the Master Agent on Another Port](#)
- [Manually Configuring the SNMP Master Agent](#)
- [Editing the Master Agent CONFIG File](#)
- [Defining sysContact and sysLocation Variables](#)
- [Configuring the SNMP Subagent](#)
- [Starting the SNMP Master Agent](#)
- [Enabling the Subagent](#)

## Starting the Master Agent on Another Port

The Administration interface will not start the SNMP master agent on ports other than 161. However, you can manually start the master agent on another port using the following steps:

1. Edit `install_dir/lib/snmp/magt/CONFIG` to specify the desired port.
2. Run the start script as follows:

```
cd instance_root/admin-server ./start -shell install_dir/lib/snmp/magt/magt
install_dir/lib/snmp/magt/CONFIG
install_dir/lib/snmp/magt/INIT
```

The master agent will then start on the desired port. However, the Administration interface will be able to detect that the master agent is running.

## Manually Configuring the SNMP Master Agent

To configure the master SNMP agent manually:

1. Log in as root.
2. Check to see if there is an SNMP daemon (`snmpd`) running on port 161.  
If an SNMP daemon is running, make sure you know how to restart it and which MIB trees it supports. Then kill its process.
3. Edit the `CONFIG` file located in `lib/snmp/magt` in the server root directory.
4. (Optional) Define `sysContact` and `sysLocation` variables in the `CONFIG` file as described in [“Defining `sysContact` and `sysLocation` Variables” on page 175](#).

## Editing the Master Agent CONFIG File

The `CONFIG` file defines the community and the manager that will work with the master agent. The manager value should be a valid system name or an IP address.

Here is an example of a basic `CONFIG` file:

```

COMMUNITY      public
                ALLOW ALL OPERATIONS

MANAGER        manager_station_name
                SEND ALL TRAPS TO PORT 162
                WITH COMMUNITY public

```

## Defining sysContact and sysLocation Variables

You can edit the `CONFIG` file to add initial values for `sysContact` and `sysLocation` which specify the `sysContact` and `sysLocation` MIB-II variables. The strings for `sysContact` and `sysLocation` in this example are enclosed in quotes. Any string that contains spaces, line breaks, tabs, and so on must be in quotes. You can also specify the value in hexadecimal notation.

Here is an example of a `CONFIG` file with `sysContact` and `sysLocation` variables defined:

```

COMMUNITY      public
                ALLOW ALL OPERATIONS

MANAGER        nms2
                SEND ALL TRAPS TO PORT 162
                WITH COMMUNITY public

INITIAL        sysLocation "Server room
901 San Antonio Road
Palo Alto CA 94303
USA"

INITIAL        sysContact "John Doe
email: jdoe@sun.com"

```

## Configuring the SNMP Subagent

To configure the SNMP subagent, perform the following steps:

1. From the Admin Server, select the server instance node in the left pane.
2. Select the Monitoring tab from the right pane.
3. Select the SNMP Subagent Configuration link.
4. (UNIX only) Enter the name and domain of the server in the Master Host field.
5. Enter the Description of the server, including operating system information.
6. Enter the Organization responsible for the server.
7. Enter the location of the server instance.
8. Enter the name of the person responsible for the server and the person's contact information in the Contact field.
9. Select On to Enable the SNMP Statistics Collection.
10. Click OK.
11. Access App Server Instances and your server instance in the left pane, then click Apply Changes.

## Starting the SNMP Master Agent

Once you have installed the SNMP master agent, you can start it manually or by using the Admin Server from the Administration interface.

### Manually Starting the SNMP Master Agent

To start the master agent manually, enter the following at the command prompt:

```
# magt CONFIG INIT&
```

The `INIT` file is a nonvolatile file that contains information from the MIB-II system group, including system location and contact information. If `INIT` doesn't already exist, starting the master agent for the first time will create it.

---

<b>NOTE</b>	An invalid manager name in the <code>CONFIG</code> file will cause the master agent start-up to fail.
-------------	---

---



To start a master agent on a non-standard port, use one of two methods:

**Method 1:** In the `CONFIG` file, specify a transport mapping for each interface over which the master agent listens for SNMP requests from managers. Transport mappings allow the master agent to accept connections at the standard port and at a non-standard port. The master agent can also accept SNMP traffic at a non-standard port. The maximum number of concurrent SNMP is limited by your target system's limits on the number of open sockets or file descriptors per process. Here is an example of a transport mapping entry:

```
TRANSPORT      extraordinary    SNMP
                OVER UDP SOCKET
                AT PORT 11161
```

After editing the `CONFIG` file manually, you should start the master agent manually by typing the following at the command prompt:

```
# magt CONFIG INIT&
```

**Method 2:** Edit the `/etc/services` file to allow the master agent to accept connections at the standard port as well as at a non-standard port.

## Starting the SNMP Master Agent Using the Admin Server

To start the SNMP master agent using the Admin Server, perform the following steps:

---

**NOTE** You must be logged in the Sun Java System Application Server as root to start the SNMP Master Agent.

---

1. Log in to the Admin Server.
2. From the Admin Server node in the left pane, choose the Monitoring tab.
3. Choose the SNMP Agent Control link near the top of the right pane.

The SNMP Agent Control page is displayed.

4. Click Start.

You can also stop and restart the SNMP master agent from the SNMP Agent Control page.

## Enabling the Subagent

After you have installed the master agent that comes with the Administration Server, you must enable the subagent for your server instance before you attempt to start it. For information on installing the master agent, see [“Installing the SNMP Master Agent” on page 172](#).

You can use the subagent to stop the SNMP function on UNIX/Linux platforms. You must stop the subagent first, then the master agent. If you stop the master agent first, you may not be able to stop the subagent. If that happens, restart the master agent, stop the subagent, then stop the master agent.

To enable the SNMP subagent:

1. Expand the App Server Instances node in the left pane.
2. Select the server instance, then click the Monitoring tab.
3. Select the SNMP Subagent Control option to display the SNMP subagent Control page.

From this page, you can start, stop, or restart the SNMP subagent. The status of the subagent is indicated just above the control buttons.

On the Windows platform, the Windows SNMP service is used for monitoring the Application Server; it can be controlled from the Control Panel/Administrative Tools/Services option.

---

<b>NOTE</b>	After making any SNMP configuration changes, you must click OK, then restart SNMP subagent from the SNMP Subagent Control page.
-------------	---

---

# Configuring J2EE Containers

Sun Java System Application Server provides various J2EE containers in compliance with the J2EE 1.3 specification. Containers provide runtime support for J2EE application components such as Enterprise Java Beans (EJBs) and Message Driven Beans (MDBs). MDBs and EJBs never interact directly with other J2EE application components. They use the protocols and methods of the EJB container for interacting with each other and with platform services, such as the Java Transaction Service. The container is interposed between application components and J2EE services. This allows the container to transparently inject the services defined by the components' deployment descriptors, such as declarative transaction management, security checks, resource pooling, and state management.

Sun Java System Application Server incorporates the Web Container and the EJB Container.

This chapter includes the following topics:

- [About the Web Container](#)
- [About the EJB Container](#)

## About the Web Container

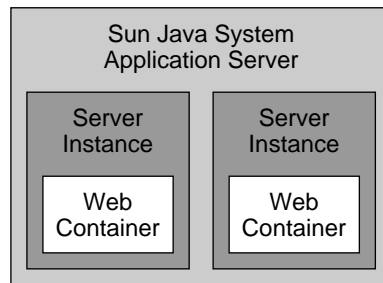
The Web Container is a J2EE container that hosts web applications. The web container extends the web server functionality by providing developers the environment to run servlets and Java Server Pages (JSPs). Servlets provide a component-based, platform-independent method for building web-based applications, without the performance limitations of CGI (Common Gateway Interface) programs. The JSP technology is an extension of the servlet technology,

created to support authoring of HTML and XML pages. Servlets or JSPs contained in a web container are capable of invoking bean methods in an Enterprise Java Beans (EJB) container. Bean methods are invoked either through local invocation or remote invocation using an Object Request Broker (ORB).

The web container also provides web applications access to local EJBs that are located using JNDI (Java Naming Directory Interface).

[Figure 7-1](#) explains the role and the location of the web container in the Sun Java System Application Server architecture:

**Figure 7-1** Web Container within the Application Server Architecture



This section covers the following topics:

- [Understanding the Web Container's Role](#)
- [Web Application Configuration](#)
- [Web Application Deployment](#)
- [Single Sign-on Facility](#)
- [Logging the Web Container](#)
- [Configuring Availability in the Web Container \(Enterprise Edition\)](#)

## Understanding the Web Container's Role

The primary role of the web container is to provide a run-time environment for web applications and provide services (database access, security, multi-threading, and so on) to web applications hosted in the container. A web application is a collection of servlets, HTML pages, classes, and other resources that make up a complete application on Sun Java System Application Server.

The following are the elements of a web application:

- Servlets
- JSP pages
- Utility classes
- Static documents (html, images, sound files, etc.)
- Client side Java applets, beans, and classes
- Descriptive meta information which ties all of the above elements together.

Web applications may be deployed in the web containers running in a Sun Java System Application Server.

For more information on how to configure and use the web server plug-in with Sun Java System Application Server, see [“Configuring the Web Server Plug-in” on page 311](#).

## Web Application Configuration

You can also configure web containers to deploy web applications within virtual servers. The web container can be configured to contain more than one virtual server. Each virtual server can be configured to host any number of web applications. Web applications are scoped within the context of a virtual server. For more information about virtual servers, see [Chapter 15, “Using Virtual Servers.”](#)

The following topics are covered in this section:

- [Virtual Server Attributes](#)
- [Web-module Attributes](#)

### Virtual Server Attributes

You can specify values for certain configurable attributes for a virtual server. A virtual server can have more than one web application associated with it. A user needs to *sign on* to a web application.

If the attribute for single sign on, `sso-enabled`, is set to the default value `true` in the `server.xml` file, a user can sign on to any one of the web applications associated with the specific virtual server. The user’s identity is then recognized by all other web applications running on the same virtual server. If the value for `sso-enabled` is set to `false`, then single sign on is disabled for all web applications in this virtual server.

The `sso-enabled` attribute is dynamically configurable and does not require a server restart to enable effect.

More information about single-sign on is provided in the section [“Single Sign-on Facility” on page 184](#).

## Web-module Attributes

The Application Server-specific deployment descriptors are specified in a file called `sun-web.xml`, which can be found in the `WEB-INF` directory of a given web application.

Usually, there is a `sun-web.xml` file configured for each web application. However, the web container does not require that every web application have a `sun-web.xml` file. In the absence of the `sun-web.xml` file, the web container assumes default values for all Application Server-specific attributes.

### *The context-root Attribute*

This attribute defines the context root at which the web application is installed. If this attribute is an empty string then this web application is designated to be the default web application for the virtual server. The default web application for a virtual server responds to all requests that cannot be resolved to other web applications deployed to the virtual server. Every virtual server has a default web application.

For the default web application, the value of this field should be an empty string `""`.

### *The location Attribute*

The input for this attribute should be a valid directory path, which indicates the location of the default web application. During the installation process, the location of the default web application is set to the `modules/default-web-app/` directory.

The `location` attribute is required and can be either a fully qualified or relative path to the directory in which the contents of the WAR (Web ARchive) file have been extracted. If the path specified is relative, then it needs to be relative to the *application root directory* defined at the virtual server level.

For example:

```
location="applications/<ear name>/<war-module name>/"
```

```
location="modules/<war-module name>"
```

```
location="/u/myapps/<war-module name>"
```

```
location="/u/myapps/<ear-name>/<war-module name>"
```

### *The enabled Attribute*

The default value for this attribute is `true`, indicating that a web application is enabled to service requests. By setting the value of the `enabled` attribute to `false`, you can temporarily disable the web application from servicing requests. However, the contents of the web application (as stored in your hard-disk) are not removed.

## Web Application Deployment

The web container deploys web applications from a Web ARchive (WAR) file or from a directory containing an exploded view (`WEB-INF/lib`, `WEB-INF/classes`, and so on) of the WAR file. You do not need to restart the server to deploy an application.

The web container deploys a “default” web application on each virtual server. The default location (directory) is in the `modules/default-web-app/` subdirectory of the `app` root directory for the virtual server. This default web application responds to all requests that cannot be resolved to other web applications deployed to the virtual server. This web application consists of an invoker servlet to handle requests to `/servlet/*` and a JSP servlet to serve JSP pages. The default web application can access EJBs as long the user denotes the EJB references in the `web.xml` and `sun-web.xml` files.

The default web application is defined in the `server.xml` of a virtual server looks as follows:

```
<web-module context-root="" location="modules/default-web-app/">
```

### Dynamic Re-deployment and Hot Deployment

Dynamic redeployment is the ability to redeploy an existing application without a server restart. Dynamic redeployment happens when an application's configuration (contents of its `xml` files) and certain classes change. Dynamic redeployment results in behavior identical to that of dynamic reloading the entire application's classes. In addition, dynamic re-deployment involves creating new application contexts (web and ejb) and getting rid of the old application contexts. Thus, dynamic re-deployment results in a brand new instance of the application (except for existing session data). This feature is also supported in development mode only and can result in exceptions similar to those for dynamic reloading. Also, configuration changes that require server restarts do not take effect until this restart happens. Dynamic reloading is activated only for applications and unshared standalone modules whose central configuration specifies it.

When a web application is reloaded, all existing session information is automatically saved and restored, regardless of whether a persistence mechanism was configured for the session manager.

Hot deployment is the ability to deploy an application at server runtime, without requiring a server restart. This feature uses the same infrastructure as is used for dynamic redeployment. However, since there is no state left over from a previous incarnation, this feature is supported at production time.

## Single Sign-on Facility

As long as the user accesses only unprotected resources in any of the web applications on a specific virtual server, the user is not challenged to authenticate himself.

When the user accesses a protected resource in any web application associated with a specific virtual server, the user will be challenged to authenticate himself using the login method defined for the web application currently being accessed.

Once authenticated, the roles associated with the user will be utilized for access-control decisions across all of the associated web applications. The user is then not required to authenticate himself individually to each web application.

When a the user logs out of a web application the user's sessions in all web applications will be invalidated. Any subsequent attempt to access a protected resource in any application will require the user to authenticate himself again.

The single sign-on feature utilizes HTTP cookies to transmit a token that associates each request with the saved user identity, so it can only be utilized in client environments that support cookies.

## Logging the Web Container

You can control the default logging behavior of the web container and any applications that are hosted in a virtual server, by setting different *log levels*. Note that the logging behavior doesn't affect the application's own logging.

Specifying a log level controls the type of messages that will be logged. For example, if you specify that only messages that bear the log level `FATAL` be logged, then the messages "higher" log level than this value will be silently ignored. Only messages logged with an explicit log level are compared to this value.

Messages logged with no explicit log level are logged unconditionally. The default behavior is to log all warning, error and fatal messages.



To set log levels for the web container, perform the following tasks:

1. In the left pane of the Administration interface, expand the Application Server instance tree to find the web container configuration you want to modify.
2. Expand the Containers tab, and select Web Container from the list of J2EE containers that are displayed.
3. Select the log level you want, from the Log Level drop-down list. For a listing of all the log levels and their definitions, see [Chapter 5, “Using Logging.”](#)
4. Click save to save your settings

To create additional properties for the web container, click Properties.

## Configuring Availability in the Web Container (Enterprise Edition)

You can enable high-availability features at the web container level. Availability features include failover of HTTP sessions. You can enable availability at the server instance level, the container level, the individual application level and the module level.

For availability to be enabled at a given level, it must be enabled at all higher levels as well. For example, to enable availability at the web container level, you must also enable it at the server instance level.

To set the availability for the web container using the Administration interface, follow these steps:

1. In the left pane, under the server instance, click Availability Service.
2. From the options next to Web Container Availability, choose enabled, disabled, or Specified by Instance.

Specified by Instance is the default value. If you choose Specified by Instance, the web container inherits the availability value from the instance. If availability is enabled for the instance, it is enabled for the web container. If availability is disabled for the instance, it is disabled for the web container. If the checkbox next to the Instance Level Availability field on this page is checked, then instance level availability is enabled.

3. Click Save.

You can also configure availability for the web container using the `asadmin` utility. Use the `asadmin set` command with the following syntax:

```
asadmin set --user admin_user [--password admin_password] [--passwordfile
password_file] [--host hostname] [--port port_number] [--secure | -s]
instance_name.web-container.availabilityEnabled=true
```

For more information about availability, see [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\).”](#)

## About the EJB Container

Enterprise Java Bean container is a runtime environment that controls the enterprise beans and provides them with important system level services. An EJB is a component that executes within an EJB container, which in turn executes within an EJB server. The following system level services are provided to the beans:

- Transaction Management
- Security
- Life cycle Management
- Remote connectivity
- Database connection pooling
- Naming Service

Enterprise beans are server components written in Java that contain business logic. The EJB Container provides the remote access to the bean. EJBs always work within the context of a container, which serves as a link between the EJBs and the server that hosts them. The EJB container enables distributed application building using your own components and components from other suppliers.

Through the EJB container, Sun Java System Application Server provides high-level transaction, state management, multi-threading, and resource pooling wrappers, thereby shielding you from having to know the low-level API details. This container provides all standard container services denoted by the 2.0 EJB Specification, and also provides additional services specific to the Sun Java System Application Server.

Passivation and activation processes are used by the container to manage bean activity to ensure scalability.

This section covers the following topics:

- [Understanding the EJB Container's Role](#)
- [Configuring the EJB Container](#)

## Understanding the EJB Container's Role

The EJB container provides the following standard services:

- **Passivation**

The process of transferring an EJB from memory to secondary storage. Passivation allows a bean's resources to be released without destroying the bean. In this way, a bean is made to be persistent, and can be recalled without the overhead of instantiation.

- **Activation**

The process of transferring an EJB from secondary storage to memory. The container contract establishes the relationship between an EJB and its container, and is completely transparent to a client. This relationship includes:

- **Life cycle**

For session beans, this includes the `javax.ejb.SessionBean` and `javax.ejb.SessionSynchronization` interface implementations. For entity beans, this includes the `javax.ejb.EntityBean` interface implementation. For message-driven beans, this includes the `javax.ejb.MessageDriven` interface implementation.

- **Session context**

A container implements the `javax.ejb.SessionContext` interface to pass services and information to a session bean instance when the bean instance is created.

- **Entity context**

A container implements the `javax.ejb.EntityContext` interface to pass services and information to an entity bean when the bean instance is created.

- **Message context**

A container implements the `javax.ejb.MDBContext` interface to pass services and information to a message-driven bean when the bean instance is created.

- Environment

A container implements `java.util.Properties` and makes these properties available to its EJBs.

- Service information

A container makes its services available to its EJBs.

The Sun Java System Application Server services include remote access, naming, security, concurrency, transaction control, and database access.

The EJB container is responsible for:

- Creating the implementation object (`EJBObject`) that allows remote connectivity.
- Creating the home implementation object that allows for creation of the `EJBObject`.
- Binding the home implementation object to the naming service so that clients can look up the home object.
- Ensuring that only authorized clients invoke the bean methods (through the `EJBObject`).
- Ensuring that business methods are invoked in the appropriate transactions.
- Managing the life cycle of the beans. Managing the life cycle of beans include:
  - Pooling the beans
  - Calling the appropriate callback methods (such as `ejbActivate/ejbPassivate`)
  - Managing a pool of database connections so that the applications use and reuse the connections more efficiently.

Actual implementation details are part of the container, based on a standard prescribed interface between a container and its EJBs. You are not required to know or to handle platform-specific implementation details. Instead, you can create generic, task-focused EJBs to be used with any vendor's products that support the EJB standard.

It is useful to understand the types of EJBs that are used by Sun Java System Application Server.

## Types of Enterprise Java Beans

An EJB is an object that represents one of the following:

- A session with a particular client, which automatically maintains state across multiple client-invoked methods.
- A persistent entity object, possibly shared among multiple clients.
- A stateless service, such as message handling.

Entity beans are primarily used to handle data access using the Java Database Connectivity (JDBC) API, while session beans provide transient application objects and perform discrete business tasks. There are three kinds of EJBs, as discussed in the following topics:

- [About Session Beans](#)
- [About Entity Beans](#)
- [About Message-driven Beans](#)

### *About Session Beans*

A session bean implements business rules or logic for a particular client request.

Session beans are intended to represent transient objects and processes, such as a single database record, a document copy for editing, or specialized business objects for individual clients. That is, a session bean is a private resource used only by the client that creates it. Because these objects are only available to a single client, session beans can maintain client-specific session information, called the *conversational* state.

For example, you might create an EJB to simulate an electronic shopping cart. Each time a user logs in to an application, the application creates the shopping cart session bean to hold purchases for that user. After the user logs out or finishes shopping, the session bean is released.

Session beans have the following characteristics:

- Session beans execute in relation to a single client.
- Session beans are relatively short lived.
- Session beans do not always survive a server crash.
- Session beans are removed if the EJB container crashes.
- Session beans also handle transaction management according to property settings. This is optional.

- Session beans update shared data in an underlying database. This is optional.
- A session bean can be either stateless or stateful.

**Stateless Session Beans.** A stateless session bean encapsulates a temporary piece of business logic needed by a specific client for a limited time span. Stateless session beans do not maintain the conversational state.

**Stateful Session Beans.** A stateful session bean is transient, but maintains a conversational state to preserve information about its contents and values between client calls. The conversational state enables the bean's container to maintain information about the session bean state and to recreate the state at a later point in program execution when needed.

### *About Entity Beans*

Entity beans commonly represent persistent data which is maintained directly in a database or accessed through an Enterprise Information System (EIS) application as an object. The server that hosts EJBs and an EJB container provides a scalable runtime environment for concurrently active entity EJBs.

A simple example of an entity bean is one defined to represent a single row in a database table, where each bean instance represents a specific row. A more complex example is an entity bean designed to represent complicated views of joined tables in a database, where, for example, each bean instance represents the contents of a single shopping cart.

Entity beans have the following characteristics:

- Entity beans provide an object view of data in the EIS resource, usually a database.
- Entity beans can be accessed by all users.
- Entity beans transparently survive server crashes.
- Entity beans use transactions that are either container-managed or bean-managed.

Entity beans represent persistent data, either as container-managed persistence or bean-managed persistence. An entity bean's persistence can either be managed by the bean or the container.

**Bean-managed persistence.** When an entity bean manages its own persistence. The bean developer implements persistence code (such as JDBC calls) directly in the EJB class methods. The possible downside is portability loss if a proprietary interface is used, and the risk in associating the bean to a specific database.

**Container-managed persistence.** When entity bean persistence is managed by the container. Because the container transparently manages the persistence state, you do not need to implement any data access code in the bean methods. Not only is this method simpler to implement, but it makes the bean fully portable without any ties to a specific database.

An entity bean that uses container-managed persistence is essentially an auto-generated (by the container) version of an entity bean that uses bean-managed persistence.

For more information on building and using Entity beans, see the *Sun Java System Application Server Developer's Guide to Enterprise JavaBeans Technology*.

## About Message-driven Beans

A message-driven bean is an EJB that allows J2EE applications to process messages asynchronously. A message-driven bean is driven by the arrival of Java Message Service messages.

From its creation until destruction, a message-driven bean instance lives in a message-driven bean container. The container provides security, transaction, concurrent processing of messages, life cycle management of the message-driven bean instances, and other services for the message-driven bean. The server that hosts EJBs and an EJB container provides a scalable runtime environment for concurrently active message-driven beans.

The Java Message Service API in the J2EE 1.3 platform specifies the following:

- Application clients, EJB components, and web components can send or synchronously receive a Java Message Service message. In addition, application clients can use Java Message Service messages asynchronously.
- The message-driven beans enable the asynchronous consumption of messages. A Java Message Service provider may optionally implement concurrent processing of messages by message-driven beans.

A message-driven bean represents a stateless service; it is essentially an asynchronous message consumer that is completely anonymous and has no client-visible identity. A message-driven bean has neither a home nor a component interface. A client accesses a message-driven bean through Java Message Service by sending messages to the Java Message Service destination (queue or topic) for which the message-driven bean class is the `MessageListener`.

Only message-driven beans can asynchronously receive messages. A session or entity beans is not permitted to be a Java Message Service `MessageListener`.

Message-driven beans have the following characteristics:

- Execute upon receipt of a single client message.
- Are asynchronously invoked.
- Are relatively short lived.
- Do not directly represent shared data in the database, but may access and update this data.
- Are removed if the EJB server crashes.
- Are stateless.
- Optionally, are transaction-aware.

## Configuring the EJB Container

You can configure the log level for the EJB container, and also enable monitoring. The EJB container handles both EJBs and MDBs. Using the Administration interface, you can configure settings for the EJBs and the MDBs that the container manages. This section covers the following topics:

- [Performing General EJB Container Configuration](#)
- [Configuring EJB Settings](#)
- [Configuring MDB Pool Settings](#)
- [Configuring Availability in the EJB Container \(Enterprise Edition\)](#)

### Performing General EJB Container Configuration

You configure the following aspects of the EJB container:

- Logging
- Monitoring
- Transaction Attributes

To set log levels for the EJB container, to enable monitoring and to set transaction attributes, perform the following tasks:

1. In the left pane of the Administration interface, open the Application Server instance tree, for the EJB container configuration you want to modify.



2. Expand the Containers tab, and select EJB Container from the list of J2EE containers that are displayed.
3. Mark the checkbox against Monitoring Enabled, to enable monitoring of the EJB Container. You've now enabled monitoring for the EJB container of this specific instance of Sun Java System Application Server. For a list of the monitorable aspects of the EJB container, see the table "[Monitoring Statistics for EJB Container.](#)"
4. Select the log level you want, from the Log Level drop-down list. For a listing of all the log levels and their definitions, [Chapter 5, "Using Logging."](#) Specifying a log level controls the type of messages that will be logged. For example, if you specify that only messages that bear the log level `FATAL` be logged, then the messages "higher" log level than this value will be silently ignored. Only messages logged with an explicit log level are compared to this value.

Messages logged with no explicit log level are logged unconditionally. The default behavior is to log all warning, error and fatal messages.

5. From the Commit Option drop down list, select the Commit Option you want to use for the EJB Container.

A transaction can end in two ways: with a commit or a rollback. When a transaction commits, the data modifications made by its statements are saved. When you design an enterprise bean, you determine if the commit is a container-managed or a bean-managed transaction. The options in the UI are therefore, B for bean-managed commit, and C for container-managed commit.

6. Click the Properties button to create new properties for the EJB container.
7. Click OK to save your settings.

The following table displays the attributes of the EJB container that can be monitored:

**Table 7-1** Monitoring Statistics for EJB Container

Statistic Name	Data Type & units	Range of values	Comments
minBeansInPool	Integer	0-MAXINT	The preferred minimum number of beans in the pool (applies to stateless session beans)
initialBeansInPool	Integer	0-MAXINT	The initial number of beans in the pool (applies to stateless session beans).
maxBeansInPool	Integer	0-MAXINT	Maximum number of beans in the pool. (applies to stateless session beans).

**Table 7-1** Monitoring Statistics for EJB Container

Statistic Name	Data Type & units	Range of values	Comments
beanIdleTimeoutInSeconds	Integer	0-MAXLONG	Idle timeout in seconds beyond which the bean will be destroyed.
numBeansCreated	Integer	0-MAXINT	Number of beans so far created.
numBeansDestroyed	Integer	0-MAXINT	Number of beans so far destroyed.
numThreadsWaiting	Integer	0-MAXINT	Number of threads waiting for free beans
numBeansInPool	Integer	0-MAXINT	Number of beans available in pool. (If this is greater than zero, then numThreadsWaiting must be 0)
maxBeansInCache	Integer	0-MAXINT	The maximum number of beans in the cache (applies to entity and stateful beans).
minBeansInCache	Integer	0-MAXINT	The preferred minimum number of beans in the cache (applies to entity and stateful beans).
cacheFaultsPercentage	Double		Number of cache misses that resulted in activating from backup store.

## Configuring EJB Settings

Using the Administration interface, you can configure the default pool and bean cache settings for the EJBs managed by the EJB container, as discussed in the following topics:

- [To Configure EJB Pool Settings](#)
- [To Configure EJB Cache Settings](#)

### *To Configure EJB Pool Settings*

To configure EJB pool settings, perform the following tasks:

1. In the left pane of the Administration interface, open the Application Server instance tree, whose EJB settings you want to modify.
2. Expand the Containers tab, and select EJB Container from the list of J2EE containers that are displayed.
3. In the Steady Pool Size field, specify the minimum number of beans in the pool. This applies to stateless session beans.
4. In the Max Pool Size drop down list, specify the maximum number of beans you want in the pool, at any given point in time. This setting applies to stateless session beans

5. In the Pool Resize Quantity field, specify the number of beans to be removed from the pool, if the beans are idle for more than the time specified in the `idle-timeout-in-seconds` tag.
6. In the Idle Timeout (secs) field, specify the period, in seconds, that a bean can remain idle. When the idle-timeout period elapses and the bean is still idle, it will be destroyed.
7. Click Save to save your changes.

### *To Configure EJB Cache Settings*

To configure EJB cache settings, perform the following tasks:

1. In the left pane of the Administration interface, open the Application Server instance tree, whose EJB settings you want to modify.
2. Expand the Containers tab, and select EJB Container from the list of J2EE containers that are displayed.
3. In the Max Cache Size field, specify the maximum number of beans to be maintained in the cache. The default value for this attribute is as specified in the `idle-timeout-in-seconds` attribute.
4. In the Cache Resize Quantity field, specify the number of beans to be selected for destruction, if the number of beans in the pool exceeds the quantity specified in the `Max Cache Size` attribute.
5. In the Removal Timeout (secs) field, specify the amount of time that a bean that is idle in the backup store can remain passivated. If a bean is not accessed by a client beyond the value specified in the `removal-timeout-in-seconds` attribute, then the bean will be removed from the backup store and therefore will not be accessible to the client.
6. From the Victim Selection Policy drop-down list, select the victim selection algorithm that must be employed to select victim beans to be removed from the pool.
7. In the Idle Timeout (secs) field, specify the period for which a bean is allowed to be idle in the cache. After this period elapses, the bean is passivated. The period for which the bean remains passivated (in the idle backup store), is controlled by the `removal-timeout-in-seconds` parameter.
8. Click Save to save your changes.

## Configuring MDB Pool Settings

Using the Administration interface, you can configure the default pool settings for the MDBs managed by the EJB container. To configure default pool settings for MDBs, perform the following tasks:

1. In the left pane of the Administration interface, open the Application Server instance tree, whose MDB container configuration you want to modify.
2. Expand the Containers tab, and select EJB Container from the list of J2EE containers that are displayed.
3. Click MDB Settings. In the `Steady Pool Size` text field, specify the minimum number of beans in the pool. This applies to stateless session beans.
4. In the `Max Pool Size` field specify the maximum number of beans you want in the pool, at any given point in time.
5. In the `Pool Resize Quantity` field, specify the number of beans to be removed from the pool, if the beans are idle for more than the time specified in the `idle-timeout-in-seconds` tag.
6. In the `Idle Timeout (secs)` field, specify the period, in seconds, that a bean can remain idle. When the idle-timeout period elapses and the bean is still idle, it will be destroyed.
7. Click Save to save your settings.

## Configuring Availability in the EJB Container (Enterprise Edition)

You can enable high-availability features at the EJB container level, which enables failover of the stateful session bean state. You can enable availability at the server instance level, the container level, the individual application level and the module level. You can also enable availability for a stateful session bean itself.

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

To set the availability for the EJB container using the Administration interface, follow these steps:

1. In the left pane, click Availability Service.

2. From the options next to EJB Container Availability, choose enabled, disabled, or Specified by Instance.

Specified by Instance is the default value. If you choose Specified by Instance, the EJB container inherits the availability value from the instance. If availability is enabled for the instance, it is enabled for the EJB container. If availability is disabled for the instance, it is disabled for the EJB container. If the checkbox next to the Instance Level Availability field on this page is checked, then instance level availability is enabled.

3. Click Save.

You can also configure availability for the EJB container using the `asadmin` utility. Use the `asadmin set` command with the following syntax:

```
asadmin set --user admin_user [--password admin_password] [--passwordfile
password_file] [--host hostname] [--port port_number] [--secure | -s]
instance_name.ejb-container.availabilityEnabled=true
```

For more information about availability used with stateful session beans, see the *Sun Java System Application Server Developer's Guide to Enterprise JavaBeans Technology*.



# Using Transaction Services

Transactions are an integral part of business. A typical business transaction involves transfer of assets between two or more involved parties. Accurate records are usually stored in one or more databases. Because this information is critical for business operations, it must be valid, current, and reliable. Transaction processing can be difficult to a novice programmer. The J2EE platform provides several abstractions that simplify development of dependable transaction processing applications. In this chapter, we will discuss J2EE transactions and transaction support in Sun Java System Application Server.

This chapter discusses Java Transactions in general, and the transaction support incorporated into Application Server in specific.

This chapter includes the following topics:

- [What Is a Transaction?](#)
- [Transactions in J2EE](#)
- [Transactional Resource Managers](#)
- [Local and Distributed Transactions](#)
- [Container-Managed Transactions](#)
- [Bean-Managed Transactions](#)
- [Transaction Service Administration](#)

## What Is a Transaction?

To emulate a business transaction, a program may need to perform several steps. A financial program, for example, might transfer funds from a checking account to a savings account, by performing the steps listed in the following pseudocode:

```
begin transaction
debit checking account
credit savings account
update history log
commit transaction
```

In the preceding pseudocode, the `begin` and `commit` statements mark boundaries of the transaction. To complete this transaction, all the three steps complete must complete successfully. If all three steps don't complete successfully, data integrity could be compromised.

This guarantee is described as *atomicity*. A transaction can end in two ways: with a `commit` or a `rollback`. When a transaction commits, the modifications made by statements within the transaction boundaries are saved and made permanent. The changes are *durable*, that is they will survive future system failures. If any statement within a transaction fails, the transaction rolls back, undoing the effects of all statements executed so far in the transaction. In the pseudocode, for example, if a disk drive crashed during the credit step, the transaction would roll back and undo the data modifications made by the debit statement.

Even if a transaction fails, data integrity would be intact because the transaction accounts still balance. This aspect of transactional behavior is known as transactional *consistency*.

The transaction service also provides *isolation*, which means that phases in a transaction cannot be observed by other applications and threads, until the transaction is committed or rolled back. Once a transaction is committed, the committed transaction can be safely observed by applications and threads.

## Transactions in J2EE

Transaction processing in J2EE involves the following five participants: Transaction Manager, Application Server, Resource Manager(s), Resource Adapter(s) and the User Application. Each of these entities contribute to reliable transaction processing, by implementing different APIs and functionalities, discussed below:

- Transaction Manager provides the services and management function required to support transaction demarcation, transactional resource management, synchronization, and transaction context propagation.



- The Application server provides the infrastructure required to support the application run-time environment which includes transaction state management.
- Resource Manager (through a resource adapter) provides the application access to resources. The resource manager participates in distributed transactions by implementing a transaction resource interface used by the transaction manager to communicate transaction association, transaction completion and recovery work. An example of such a resource manager is a relational database server.
- A Resource Adapter is a system level software library that is used by the application server or client to connect to a Resource Manager. A Resource Adapter is typically specific to Resource Manager. It is available as a library and is used within the address space of the client using it. An example of such a resource adapter is the JDBC driver.
- A transactional User Application developed to operate in a J2EE application server environment, looks up transactional data sources, and optionally, the transaction manager, using JNDI. May use declarative transaction attribute settings for EJBs or explicit programmatic transaction demarcation.

The term Resource Manager is often used interchangeably with Resource Adapter, as there is a close tie between the two entities.

## Transactional Resource Managers

The following transactional resource managers are supported in J2EE transactions.

- [Databases](#)
- [JMS Providers](#)
- [J2EE Connectors](#)

### Databases

Databases are the most commonly encountered transactional resource managers in J2EE applications. JDBC is the API is used by J2EE components to access databases. Database resources are configured as JDBC resources. JDBC resources are managed by a resource managers, or JDBC Drivers. A JDBC driver may provide support for Local transactions or Global transactions, and in some cases for both local and global transactions.

Application Server supports usage of JDBC and Transactions from various J2EE components. For more details on how JDBC resources are registered and configured, see [“About JDBC Resources” on page 241](#). The Application Server is responsible for providing transaction continuity (that is, initiating a transaction and accessing the database from multiple application components). For example, a servlet may start a transaction, access a database, invoke an enterprise bean that accesses the same database as part of the same transaction, and finally, commit the transaction.

## JMS Providers

JMS stands for Java Message Service. A JMS Provider is the J2EE term for the Message Broker Service. The JMS API provides reliable and transactional exchange of messages between applications. Support of transactional JMS data sources is a required capability in J2EE. JMS resources and JDBC resources can participate in the same transaction.

Application Server comes integrated with Sun Java System Message Queue, a fully capable JMS provider and the corresponding transactional resource manager. In this manner, Application Server enables transactional JMS access from servlets, JSP pages and enterprise beans. It is also possible to use third party JMS Providers with Application Server. For more details, see [Chapter 10, “Using the JMS Service.”](#)

## J2EE Connectors

Application Server supports resource adapters that use XATransaction mode as transaction resource managers. The platform must enable transactional access to the resource adapter from servlets, JSP pages, and enterprise beans. It is possible to access the resource adapter from multiple application components within a single transaction. For example, a servlet may wish to start a transaction, access the resource adapter, invoke an enterprise bean that also accesses the resource adapter as part of the same transaction, and finally, commit the transaction.

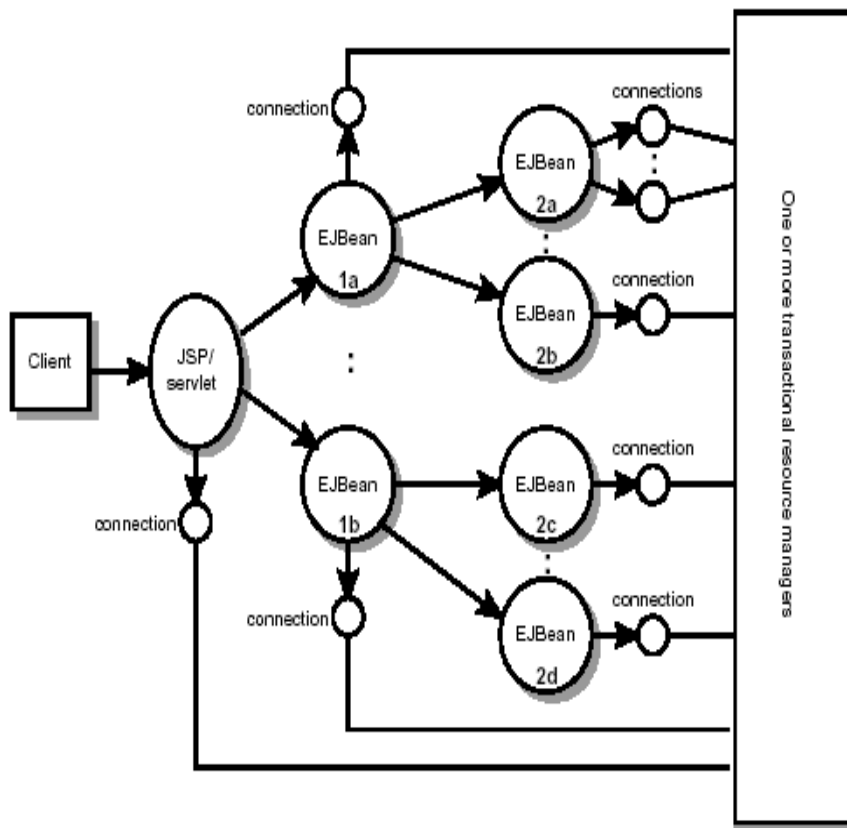
# Local and Distributed Transactions

A transaction that involves only one resource can be completed using local transactions. A local transaction also requires that all participating application components execute within one process. Transactions that involve more than one resource, or multiple participant processes become distributed or global transactions. Local transaction optimization uses the resource manager specific optimization and it is transparent to the J2EE application.

The type of transaction is largely determined by the interfaces implemented of the involved resource managers. For example, a JDBC Data Source that implements `javax.sql.DataSource` interface can participate in Local transactions. A Data Source that implements `javax.sql.XADataSource` is capable of taking part in global transactions. Some JDBC resources implement both the interfaces, and when such a JDBC resource is registered with Application Server, it may be necessary to provide additional configuration information in the Application Server configuration to indicate the preferred capability for that resource.

Local transactions are simpler and naturally more efficient than global transactions. Local transactions are inadequate when the data that needs to be transformed is spread across multiple data sources. Sometimes, it is not possible to predict how many data sources would need to be enlisted in a transaction. So global transactions are encountered quite frequently in the real world. Some performance enhancing optimizations are possible with global transactions.

J2EE supports transactional applications comprising any combination of servlets or JSPs accessing multiple enterprise beans within a transaction. Each component may acquire one or more connections to access one or more transactional resource managers. In the following figure, the call tree starts from a servlet or JSP page accessing multiple enterprise beans, which in turn may access other enterprise beans. The components access resource managers via connections.

**Figure 8-1** J2EE Components Accessing Resources in a Transaction

For example, an application may require that all the components in the above figure access resources as part of a single transaction. The application server provider must provide the transaction capabilities to support such a scenario.

J2EE transactional management supports flat transactions. A flat transaction cannot have any child (nested) transactions.

Transaction Recovery is an important aspect of distributed transactions. When a resource becomes unreachable during critical points, or if there are other unrecoverable errors, the status of the distributed transaction can be in question. Automatic and manual recovery of stranded/incomplete transactions is an

important feature in Sun Java System Application Server. You can enable automatic transaction recovery by using the Administration interface. For more information on how to control transaction recovery, see [“Transaction Service Administration” on page 213](#).

Connections—used as a synonym here for resources—can be marked as either shareable or non-shareable. A J2EE application component that intends to use a connection in an un-shareable way must provide deployment information to that effect, to prevent the connection from being shared by the container. Examples of when this may be needed include situations with changed security attributes, isolation levels, character settings, and localization configuration.

Containers must not attempt to share connections that are marked un-shareable. If a connection is not marked as unshareable, it must be transparent to the application whether the connection is actually shared or not.

J2EE application components may use the optional deployment descriptor element `res-sharing-scope` to indicate whether a connection to a resource manager is shareable or unshareable. Containers should assume connections to be shareable if no deployment hint is provided. J2EE application components may cache connection objects and reuse them across multiple transactions. Containers that provide connection sharing should transparently switch such cached connection objects (at dispatch time) to point to an appropriate shared connection with the correct transaction scope.

When designing an enterprise bean application, the developer must determine how the boundaries are specified.

## Container-Managed Transactions

In an enterprise bean with container-managed transactions, the EJB container sets the boundaries of the transactions. You can use container-managed transactions with any type of enterprise bean: session, entity, or message-driven. Container-managed transactions simplify development because the enterprise bean code does not explicitly mark the transaction's boundaries. The code does not include statements that begin and end the transaction.

Typically, the container begins a transaction immediately before an enterprise bean method starts. It commits the transaction just before the method exits. Each method can be associated with a single transaction. Nested or multiple transactions are not allowed within a method.

Container-managed transactions do not require all methods to be associated with transactions. When deploying a bean, you specify which of the bean's methods are associated with transactions by setting the transaction attributes.

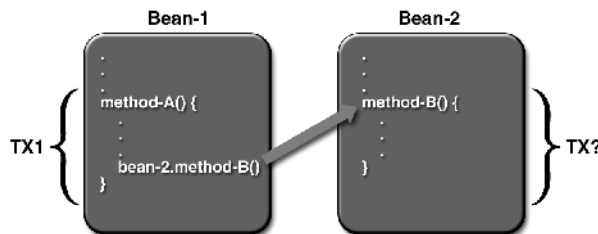
This section covers the following topics:

- [Transaction Attributes](#)
- [Setting Transaction Attributes](#)
- [Rolling Back a Container-Managed Transaction](#)
- [Synchronizing a Session Bean's Instance Variables](#)
- [Methods Not Allowed in Container-Managed Transactions](#)

## Transaction Attributes

A transaction attribute controls the scope of a transaction. The following figure illustrates why controlling the scope is important. In the diagram, method-A begins a transaction and then invokes method-B of Bean-2. When method-B executes, does it run within the scope of the transaction started by method-A or does it execute with a new transaction? The answer depends on the transaction attribute of method-B.

**Figure 8-2** Transaction Attributes



A transaction attribute may have one of the following values:

- [Required](#)
- [RequiresNew](#)
- [Mandatory](#)
- [NotSupported](#)

- Supports
- Never

## Required

If the client is running within a transaction and invokes the enterprise bean's method, the method executes within the client's transaction. If the client is not associated with a transaction, the container starts a new transaction before running the method.

The `Required` attribute will work for most transactions. Therefore, you may want to use it as a default, at least in the early phases of development. Because transaction attributes are declarative, you can easily change them at a later time.

## RequiresNew

If the client is running within a transaction and invokes the enterprise bean's method, the container takes the following steps:

- Suspends the client's transaction
- Starts a new transaction
- Delegates the call to the method
- Resumes the client's transaction after the method completes

If the client is not associated with a transaction, the container starts a new transaction before running the method.

You should use the `RequiresNew` attribute when you want to ensure that the method always runs within a new transaction.

## Mandatory

If the client is running within a transaction and invokes the enterprise bean's method, the method executes within the client's transaction. If the client is not associated with a transaction, the container throws the

`TransactionRequiredException`.

Use the `Mandatory` attribute if the enterprise bean's method must use the transaction of the client.

## NotSupported

If the client is running within a transaction and invokes the enterprise bean's method, the container suspends the client's transaction before invoking the method. After the method has completed, the container resumes the client's transaction.

If the client is not associated with a transaction, the container does not start a new transaction before running the method.

Use the `NotSupported` attribute for methods that don't need transactions. Because transactions involve overhead, this attribute may improve performance.

## Supports

If the client is running within a transaction and invokes the enterprise bean's method, the method executes within the client's transaction. If the client is not associated with a transaction, the container does not start a new transaction before running the method.

Because the transactional behavior of the method may vary, you should use the `Supports` attribute with caution.

## Never

If the client is running within a transaction and invokes the enterprise bean's method, the container throws a `RemoteException`. If the client is not associated with a transaction, the container does not start a new transaction before running the method.

## Attribute Summary

The following table summarizes the effects of the transaction attributes. Both the T1 and T2 transactions are controlled by the container. A T1 transaction is associated with the client that calls a method in the enterprise bean. In most cases, the client is another enterprise bean. A T2 transaction is started by the container just before the method executes.

In the last column of, the term "None" indicates that the business method does not execute within a transaction controlled by the container. However, the database calls in such a business method might be controlled by the transaction manager of the DBMS.



**Table 8-1** Transaction Attributes

Transaction Attribute	Client's Transaction	Business Method's Transaction
Required	None	T2
	T1	T1
RequiresNew	None	T2
	T1	T2
Mandatory	None	error
	T1	T1
NotSupported	None	None
	T1	None
Supports	None	None
	T1	T1

## Setting Transaction Attributes

Because transaction attributes are stored in the deployment descriptor, they can be changed during several phases of J2EE application development: enterprise bean creation, application assembly, and deployment. However it is the responsibility of a developer to specify the attributes when creating the bean. The attributes should be modified only by an application developer who is assembling components into larger applications. The individual who deploys the J2EE application is not responsible for specifying the transaction attributes.

You can specify the transaction attributes for the entire enterprise bean or for individual methods. If you've specified one attribute for a method and another for the bean, the attribute for the method takes precedence. When specifying attributes for individual methods, the requirements differ with the type of bean. Session beans need the attributes defined for business methods, but do not allow them for the create methods. Entity beans require transaction attributes for the business, create, remove, and finder methods. Message-driven beans require transaction attributes (either `Required` or `NotSupported`) for the `onMessage` method.

## Rolling Back a Container-Managed Transaction

There are two ways to roll back a container-managed transaction. First, if a system exception is thrown, the container will automatically roll back the transaction. Second, by invoking the `setRollbackOnly` method of the `EJBContext` interface, the bean method instructs the container to roll back the transaction. If the bean throws an application exception, the rollback is not automatic, but may be initiated by a call to `setRollbackOnly`.

In the following example, the `transferToSaving` method of the `BankEJB` example illustrates the `setRollbackOnly` method. If a negative checking balance occurs, `transferToSaving` invokes `setRollbackOnly` and throws an application exception (`InsufficientBalanceException`). The `updateChecking` and `updateSaving` methods update database tables. If the updates fail, these methods throw a `SQLException` and the `transferToSaving` method throws an `EJBException`. Because the `EJBException` is a system exception, it causes the container to automatically roll back the transaction. Here is the code for the `transferToSaving` method:

```
public void transferToSaving(double amount) throws
    InsufficientBalanceException {

    checkingBalance -= amount;
    savingBalance += amount;

    if (checkingBalance < 0.00) {
        context.setRollbackOnly();

        throw new InsufficientBalanceException();
    }

    try {
        updateChecking(checkingBalance);

        updateSaving(savingBalance);
    } catch (SQLException ex) {
        throw new EJBException
            ("Transaction failed due to SQLException: "
             + ex.getMessage());
    }
}
```

When the container rolls back a transaction, it always undoes the changes to data made by SQL calls within the transaction. However, only in entity beans will the container undo changes made to instance variables. (It does so by automatically invoking the entity bean's `ejbLoad` method, which loads the instance variables from

the database.) When a rollback occurs, a session bean must explicitly reset any instance variables changed within the transaction. The easiest way to reset a session bean's instance variables is by implementing the `SessionSynchronization` interface.

You can also roll back a transaction by passing the transaction ID through the command line interface. For more details, please see [“Administering Transactions Using the Command-Line Interface” on page 216](#).

## Synchronizing a Session Bean's Instance Variables

The `SessionSynchronization` interface, which is optional, allows you to synchronize the instance variables with their corresponding values in the database. The container invokes the `SessionSynchronization` methods—`afterBegin`, `beforeCompletion`, and `afterCompletion`—at each of the main stages of a transaction.

The `afterBegin` method informs the instance that a new transaction has begun. The container invokes `afterBegin` before it invokes the first business method within a transaction. The `afterBegin` method is a good place to load the instance variables from the database. The `BankBean` class, for example, loads the `checkingBalance` and `savingBalance` variables in the `afterBegin` method:

```
public void afterBegin() {

    System.out.println("afterBegin()");
    try {
        checkingBalance = selectChecking();
        savingBalance = selectSaving();
    } catch (SQLException ex) {
        throw new EJBException("afterBegin Exception: " +
                                ex.getMessage());
    }
}
```

The container invokes the `beforeCompletion` method after the business method has finished, but just before the transaction commits. The `beforeCompletion` method is the last opportunity for the session bean to roll back the transaction (by calling `setRollbackOnly`). If it hasn't already updated the database with the values of the instance variables, the session bean may do so in the `beforeCompletion` method.

The `afterCompletion` method indicates that the transaction has completed. It has a single boolean parameter, whose value is true if the transaction was committed and false if it was rolled back. If a rollback occurred, the session bean can refresh its instance variables from the database in the `afterCompletion` method:

```
public void afterCompletion(boolean committed) {

    System.out.println("afterCompletion: " + committed);
    if (committed == false) {
        try {
            checkingBalance = selectChecking();
            savingBalance = selectSaving();
        } catch (SQLException ex) {
            throw new EJBException("afterCompletion SQLException:
                " + ex.getMessage());
        }
    }
}
```

## Methods Not Allowed in Container-Managed Transactions

You should not invoke any method that might interfere with the transaction boundaries set by the container. The list of prohibited methods follows:

- The `commit`, `setAutoCommit`, and `rollback` methods of `java.sql.Connection`
- The `getUserTransaction` method of `javax.ejb.EJBContext`
- Any method of `javax.transaction.UserTransaction`

You may, however, use these methods to set boundaries in bean-managed transactions.

# Bean-Managed Transactions

In a bean-managed transaction, the code in the session or message-driven bean explicitly marks the boundaries of the transaction. An entity bean cannot have bean-managed transactions; it must use container-managed transactions instead. Although beans with container-managed transactions require less coding, they have one limitation: When a method is executing, it can be associated with either a single transaction or no transaction at all. If this limitation will make coding your bean difficult, you should consider using bean-managed transactions.

The following pseudocode illustrates the kind of fine-grained control you can obtain with bean-managed transactions. By checking various conditions, the pseudocode decides whether to start or stop different transactions within the business method.

```
begin transaction
...
update table-a
...
if (condition-x)
    commit transaction
else if (condition-y)
    update table-b
    commit transaction
else
    rollback transaction
    begin transaction
    update table-c
    commit transaction
```

## Transaction Service Administration

You can administer transaction by using the Administration interface, or by using the Command Line Interface.

This section covers the following topics:

- [Administering Transactions Using the Administration Interface](#)
- [Administering Transactions Using the Command-Line Interface](#)

# Administering Transactions Using the Administration Interface

Using the Administration interface, you can enable monitoring, set log levels, and specify advanced options for your transactions.

You can control instance-wide transaction service attributes, such as recovery policy and time-outs. The properties and configuration that you specify here are stored in the `server.xml` file.

To configure transaction service options, perform the following tasks:

- 1. In the left pane of the Administration interface, open the Application Server instance tree, whose transaction configuration you want to modify.
- 2. Select Transaction Service from the list of J2EE services that are displayed.
- 3. To enable monitoring for your transactions, mark the Monitoring Enabled checkbox. The following table lists the features of Java Transaction services that can be monitored:

**Table 8-2** Monitorable Attributes of Java Transaction Services

Property	Type	Description
transactionsCompleted	int	Number of transactions completed since monitoring is enabled
transactionsRolledBack	int	Number of transactions rolled back since monitoring is enabled
transactionsRecovered	int	Number of transactions recovered since monitoring is enabled
transactionsInFlight	int	Number of transactions presently being processed
timeStamp	long	In milliseconds, recording the time at which the statistic was produced. This will be whatever that is reported by <code>System.currentTimeMillis()</code>

- 4. Select the log level you want to set for your transactions, from the Log Level drop-down list. For a more information on log levels and how they are incorporated, see [Chapter 5, “Using Logging.”](#)

5. Mark the checkbox against Recover on Restart to automatically recover failed transactions at server restart. When a resource becomes unreachable during critical points in the transaction Commit protocol, transactions may not complete and remain in transaction log file. If this check box has been marked, the server attempts to recover stranded transaction upon server restart. If the involved resources remain unreachable, this may delay server restart. This checkbox is not marked by default.
6. For enterprise beans with container-managed transactions, you can control the transaction timeout interval by setting the value of the Transaction Timeout (secs) property.

If the value of this property is set to 0, your transaction does not time out.

In the Transaction Timeout (secs) field, specify the transaction timeout interval. If a transaction is not completed within the specified time, the transaction will be rolled back. If the value set for this attribute is 0, the transaction does not time out.

7. In the Transaction Log Location field, specify the absolute path to the directory in which you want to store your log files. You need to restart your server for the new transaction log directory to be effective.
8. From the Heuristic Decision drop-down box, select the heuristic decision that you want to apply to your transactions. Select Commit or Rollback from the options indicated, to specify how the Application Server should determine the outcome of an in-doubt transaction during recovery, if the outcome cannot be determined unambiguously. If the Heuristic Decision is set to Rollback, it rolls back the transaction. In some cases, it may be acceptable to commit such a transaction.
9. In the Keypoint Interval (transactions) field, specify the number of transactions between keypoint operations in the log. Keypoint operations reduce the size of the transaction log file by removing entries for completed transactions and compressing the file. A larger value for this attribute results in a larger transaction log file, but fewer keypoint operations mean a potentially better performance. A smaller value (for example, 100) results in smaller log files, but slightly reduced performance due to the greater frequency of keypoint operations.

# Administering Transactions Using the Command-Line Interface

You can administer and monitor database transactions using the Command Line Interface (CLI), as explained in the following sections:

- [Listing In-Flight Transactions](#)
- [Managing Transactions](#)
- [Freezing the Transaction Service](#)
- [Monitoring Transactions](#)

These sections explain how to use the Command Line Interface to manage and monitor transactions.

## Listing In-Flight Transactions

The following command should be used to get the in-flight transaction data (assuming that you are in multimode and have already set the user name and password):

```
- asadmin> get --monitor <instanceName>.transaction-service.inflight-tx
```

The multi-line output will look like this:-

```
Transaction Id State Elapsed Time (ms)
txnid1 Prepared 20
txnid2 Active 100
txnid3 Active 120
... ..
```

## Managing Transactions

In the example given in “[Listing In-Flight Transactions](#)” on page 216, let us assume that you want to rollback transactions with the transaction Ids: txn-ids, txnid2, and txnid3. A sample command to rollback chosen transactions, would look like the following example:

```
asadmin> set --monitor
<instanceName>.transaction-service.rollback-list=txnid2,txnid3
```



## Freezing the Transaction Service

To freeze the transaction service, run the following command:

```
asadmin> set --monitor <instanceName>.transaction-service.freeze=true
```

When the transaction service is frozen the transaction manager in the application server, will suspend all in-flight transactions. Freezing is not recommended on a production deployment system.

To unfreeze the transaction service, run the following command:

```
asadmin> set --monitor <instanceName>.transaction-service.freeze=false
```

When the transaction service is set in motion again, the system continues where it left off. If a live system was left in frozen state for too long, some database connections may time out, resulting in rolled back transactions.

## Monitoring Transactions

To get the monitoring data of transactions, including in-flight transaction data, run the following command:

```
asadmin> get --monitor <instanceName>.transaction-service.*
```

If there are no active transactions when you run this command, you will get the following output:

```
total-tx-completed = 5
total-tx-rolledback = 2
total-tx-inflight = 0
isFrozen = false
tx-inflight = No active transactions found.
```

If there are active transactions, when you run this command, you will get the following output:

```
total-tx-completed = 5
total-tx-rolledback = 2
total-tx-inflight = 2
isFrozen = false
tx-inflight =
Transaction Id State Elapsed Time(ms)
```

txnidl Prepared 500

txnidl Active 360

# Configuring Naming and Resources

This chapter describes the J2EE Resources used by Sun Java System Application Server and discusses the methods used to create and manage these resources:

This chapter includes the following topics:

- [About J2EE Naming Services and Resources](#)
- [About Java Naming and Directory Interface \(JNDI\)](#)
- [About Persistence Manager Resources](#)
- [About JDBC Resources](#)
- [About Java Mail Resources](#)

## About J2EE Naming Services and Resources

J2EE applications including EJBs, web application components and application clients may access a wide variety of resources such as resource managers, data sources (for example SQL datasources), connection factories, mail sessions, Java Message Service destination objects, and URL connection factories. The J2EE platform exposes such resources to the applications via Java Naming and Directory (JNDI) naming service.

Sun Java System Application Server allows you to create and manage the following J2EE resources:

- [JDBC Datasources](#)
- [Java Mail Sessions](#)
- [JMS Destinations](#)

## JDBC Datasources

A JDBC Datasource is a J2EE resource that you can create and manage using Sun Java System Application Server.

The JDBC API is the API for connectivity with relational database systems. The JDBC API has two parts:

- An application-level interface used by the application components to access databases.
- A service provider interface to attach a JDBC driver to the J2EE platform.

A JDBC `DataSource` object is the representation of a data source in the Java programming language. In basic terms, a data source is a facility for storing data. It can be as sophisticated as a complex database for a large corporation or as simple as a file with rows and columns. A JDBC Datasource is a J2EE resource that can be created and managed via Sun Java System Application Server.

For more information on JDBC Datasources, see [“About JDBC Resources” on page 241](#).

## Java Mail Sessions

JMS destinations are J2EE resources that can be created and managed via Sun Java System Application Server.

Many internet applications require the ability to send email notifications, so the J2EE platform includes the JavaMail API along with a JavaMail service provider that allows an application component to send internet mail. The JavaMail API has two parts:

- An application-level interface used by the application components to send mail
- A service provider interface used at the J2EE SPI level.

Java Mail Sessions are J2EE resources that can be created and managed via Sun Java System Application Server. For more information on Java Mail Sessions, see [“About Java Mail Resources” on page 262](#).

## JMS Destinations

Java Messaging Service (JMS) is a standard API for messaging that supports reliable point-to-point messaging as well as the publish-subscribe model. This specification requires a JMS provider that implements both point-to-point messaging and publish-subscribe messaging.

JMS provides for two general types of administered objects: connection factories and destinations. While both encapsulate provider-specific information, they have very different uses within a JMS client. A connection factory is used to create connections to a message server, and destination objects are used to identify physical destinations used by the JMS messaging service.

## About Java Naming and Directory Interface (JNDI)

This section discusses the Java Naming and Directory Interface (JNDI). Java Naming and Directory Interface (JNDI) is an application programming interface (API) for accessing different kinds of naming and directory services. J2EE components locate objects by invoking the JNDI lookup method.

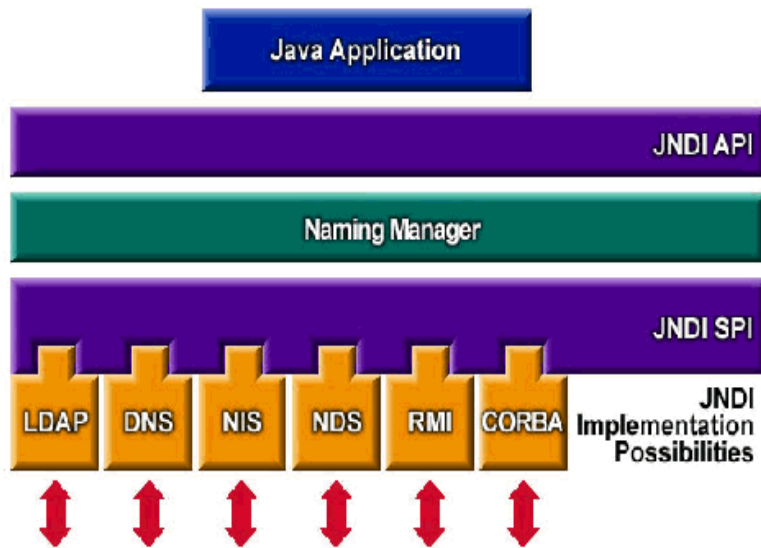
This section covers the following topics:

- [JNDI Architecture](#)
- [J2EE Naming Services](#)
- [Naming References and Binding Information](#)
- [Naming References in J2EE Standard Deployment Descriptor](#)
- [JNDI Connection Factories](#)

## JNDI Architecture

The JNDI architecture consists of an Application Programmer's Interface (API) and a Service Provider Interface (SPI). Java applications use the JNDI API to access a variety of naming and directory services. The SPI enables a variety of naming and directory services to be plugged in transparently, thereby allowing the Java application using the JNDI API to access their services. [Figure 9-1](#) illustrates the services that can be accessed through the JNDI API:

**Figure 9-1** Overview of the JNDI Architecture



## J2EE Naming Services

A JNDI name is a people-friendly name for an object. These names are bound to their objects by the naming and directory service that is provided by a J2EE server. Because J2EE components access this service through the JNDI API, we usually refer to an object's people-friendly name as its JNDI name. For example, the JNDI name of the Pointbase database is `jdbc/Pointbase`. When it starts up, Sun Java System Application Server reads information from configuration file and automatically adds JNDI database names to the name space.

J2EE application clients, enterprise beans, and web components are required to have access to a JNDI naming environment.

The application component's naming environment is a mechanism that allows customization of the application component's business logic during deployment or assembly. Use of the application component's environment allows the application component to be customized without the need to access or change the application component's source code.

A J2EE container implements the application component's environment, and provides it to the application component instance as a JNDI naming context. The application component's environment is used as follows:

- The application component's business methods access the environment using the JNDI interfaces. The application component provider declares in the deployment descriptor all the environment entries that the application component expects to be provided in its environment at runtime.
- The container provides an implementation of the JNDI naming context that stores the application component environment. The container also provides the tools that allow the deployer to create and manage the environment of each application component.
- A deployer uses the tools provided by the container to initialize the environment entries that are declared in the application component's deployment descriptor. The deployer can set and modify the values of the environment entries.
- The container makes the environment naming context available to the application component instances at runtime. The application component's instances use the JNDI interfaces to obtain the values of the environment entries.

Each application component defines its own set of environment entries. All instances of an application component within the same container share the same environment entries. Application component instances are not allowed to modify the environment at runtime. For more information on how J2EE containers such as the Web Container and EJB Containers use the JNDI naming service to look up objects, please see [“Configuring J2EE Containers” on page 179](#).

## Naming References and Binding Information

A resource reference is an element in a deployment descriptor that identifies the component's coded name for the resource. More specifically, the coded name references a connection factory for the resource. In the example given in the following section, the resource reference name is `jdbc/SavingsAccountDB`.

The JNDI name of a resource and the name of the resource reference are not the same. This approach to naming requires that you map the two names before deployment, but it also decouples components from resources. Because of this de-coupling, if at a later time the component needs to access a different resource, you don't have to change the name in the code. This flexibility also makes it easier for you to assemble J2EE applications from preexisting components.

[Table 9-1](#) lists JNDI lookups and their associated references for the J2EE resources used by Sun Java System Application Server.

**Table 9-1** JNDI Lookups and Their Associated References

JNDI Lookup Name	Associated Reference
java:comp/env	Application environment entries
java:comp/env/jdbc	JDBC DataSource resource manager connection factories
java:comp/env/ejb	EJB References
java:comp/UserTransaction	UserTransaction references
java:comp/env/mail	JavaMail Session Connection Factories
java:comp/env/url	URL Connection Factories
java:comp/env/jms	JMS Connection Factories and Destinations
java:comp/ORB	ORB instance shared across application components

## Naming References in J2EE Standard Deployment Descriptor

A naming reference is a string used by the application to look up an object in the given naming context. For each J2EE application, there is a naming context and the references are configured in the standard component deployment descriptors. This section describes the standard deployment descriptor features used in Sun Java System Application Server. This section covers the following topics:

- [Application Environment Entries](#)
- [EJB References](#)
- [References to Resource Manager Connection Factories](#)
- [Resource Environment References](#)
- [UserTransaction References](#)
- [COSNaming Service](#)



## Application Environment Entries

Environment entries, defined using `<env-entry>`, provide a way of specifying deployment time parameters to J2EE applications. Note that in case of web applications, the servlet context initialization parameters could be defined using `<context-param>`, but `<env-entry>` is the preferred way because application deployers to configure such applications parameters by explicitly specifying the name, type and values for them.

The following sample describes the syntax of `<env-entry>` as specified in the J2EE standard deployment descriptors:

```
<env-entry>
<description> Send pincode by mail </description>
<env-entry-name> mailPincode </env-entry-name>
<env-entry-value> false </env-entry-value>
<env-entry-type> java.lang.Boolean </env-entry-type>
</env-entry>
```

The `<env-entry-type>` tag specifies a fully qualified class name for the entry. Here is a code snippet to lookup the `<env-entry>` using JNDI from an application component (a term referring to a servlet/JSP or an entity bean or an IIOP application client):

```
Context initContext = new InitialContext();
Boolean mailPincode = (Boolean)
initContext.lookup("java:comp/env/mailPincode");

// one could use relative names into the sub-context
Context envContext = initContext.lookup("java:comp/env");
Boolean mailPincode = (Boolean)
envContext.lookup("mailPincode");
```

## EJB References

Apart from deployment descriptor support, the JNDI naming service enables applications to use “logical” names (called EJB references) to map to the home interfaces of enterprise beans, as described in the following example:

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<ejb-ref-type> Entity </ejb-ref-type>
<home> com.wombat.empl.EmployeeRecordHome </home>
<remote> com.wombat.empl.EmployeeRecord </remote>
<ejb-link> EmployeeEJB </ejb-link>
</ejb-ref>
```

An application component such as a JSP could access the EJB home object using JNDI, as described in the following example:

```
Context initContext = new InitialContext();
Context envContext = initContext.lookup("java:comp/env");
Object result = envContext.lookup("ejb/EmplRecord");
EmployeeRecordHome emplRecordHome = (EmployeeRecordHome)
javax.rmi.PortableRemoteObject.narrow(result, EmployeeRecordHome.class);
```

The `ejb-ref-name` element defines the string used in the application code (as in the above given example). The `ejb-link` element links this reference to target enterprise bean defined using the `ejb-name` element of the entity bean defined in the `ejb-jar.xml`. It is also possible to provide the linkage without modifying either the application deployment descriptor or the enterprise bean descriptor.

## References to Resource Manager Connection Factories

A factory is an object that creates other objects on demand. A resource factory creates resource objects, such as database connections or message service connections. They are configured using `<resource-ref>` element in the standard deployment descriptors.

The following examples describe the use of factories:

### Example A:

Declaration of a reference to a JDBC connection factory that returns objects of type `javax.sql.DataSource`:

```
<resource-ref>
<description> Primary database </description>
<res-ref-name> jdbc/primaryDB </res-ref-name>
<res-type> javax.sql.DataSource </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

### Example B:

Here is an example reference to JavaMail Session resource factory:

```
<resource-ref>
<description> mail Session </description>
<res-ref-name> mail/Session </res-ref-name>
<res-type> javax.mail.Session </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

`<res-type>` is a fully-qualified class name of the resource factory. The `<res-auth>` variable can be assigned either `Container` or `Application` as a value. For more information about configuring Java Mail session resource factories, see [“About Java Mail Resources” on page 262](#).

If `Container` is specified, the web container handles the authentication before binding the resource factory to JNDI lookup registry. If `Application` is specified, the servlet must handle authentication programmatically. Different resource factories are looked up under a separate sub-context that describes the resource type, follows:

- `jdbc/` for a JDBC `javax.sql.DataSource` factory
- `jms/` for a JMS `javax.jms.QueueConnectionFactory` or `javax.jms.TopicConnectionFactory`
- `mail/` for a JavaMail `javax.mail.Session` factory
- `url/` for a `java.net.URL` factory

Here is a code snippet to get JDBC connection from an application component with the container handling the authentication:

```
InitialContext initContext = new InitialContext();
DataSource source =
(DataSource) initContext.lookup("java:comp/env/jdbc/primaryDB");
Connection conn = source.getConnection();
```

Please note that in order to ensure that for these resource references work, the `res-ref-name` must map to valid resource factory at runtime.

## Resource Environment References

Resource environment references provide a way of accessing, via JNDI lookups, administered objects associated with a resource. For example, an application may need to access a JMS Destination object. The `<resource-env-ref>` element, defined in the standard deployment descriptors lets applications declare the resource requirements.

The main difference between `<resource-env-ref>` and `<resource-ref>` element is the absence of specific resource authentication requirement; both these elements have to be backed up by a resource factory descriptor.

**Example:**

```
<resource-env-ref>
<description> My Topic </description>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<res-env-ref-type> javax.jms.Topic </res-type>
</resource-env-ref>
```

The following piece of code allows you to access a JMS Topic object:

```
InitialContext initContext = new InitialContext();
javax.jms.Topic myTopic =
    (javax.jms.Topic) initContext.lookup("java:comp/env/jms/MyTopic");
```

Note that in order these `resource-env-ref` variables to work, the administrators will have to make target resource factories available at run-time. For more information about accessing JMS Topics and Queue destinations, see [Chapter 10, “Using the JMS Service.”](#)

## UserTransaction References

J2EE requires that containers provide a `UserTransaction` object implementation at the JNDI name `java:comp/UserTransaction`. A `UserTransaction` object lets applications to start, commit and abort transactions.

To programmatically initiate and perform transactions, components obtain reference to the container's default transaction co-ordinator, by doing a JNDI lookup for `java:comp/UserTransaction`. The returned object implements `javax.transaction.UserTransaction` interface and can be used in the program to begin, commit, rollback and query status of transactions. JNDI implementation in Sun Java System Application Server supports such lookup of the transaction co-ordinator. For more information about the `javax.transaction.UserTransaction` interface, see [“Using Transaction Services” on page 199](#).

## Initial Naming Context

The naming support in Sun Java System Application Server is based primarily on J2EE 1.3, with a few added enhancements. When an application component creates the initial context, via `InitialContext()`, Sun Java System Application Server returns an object that serves as a handle to the application's naming environment. This object in turn provides sub-contexts for the `java:comp/env` namespace. Each application gets its own namespace, that is, `java:comp/env` name space is per application and objects bound in one application's namespace don't collide with objects bound in other applications.

## COSNaming Service

The EJB interoperability protocol requires the use of the COSNaming protocol for lookup of EJB objects using the JNDI API.

EJB containers are required to be able to publish `EJBHome` object references in a CORBA `CosNaming` service. The `CosNaming` service must implement the IDL interfaces in the `CosNaming` module defined, and allow clients to invoke the resolve and list operations over IIOP.

The `CosNaming` service must follow the requirements in the CORBA Interoperable Name Service specification for providing the host, port and object key for its root `NamingContext` object. The `CosNaming` service must be able to service IIOP invocations on the root `NamingContext` at the advertised host, port and object key.

Client containers (that is, EJB, web or application client containers) are required to include a JNDI `CosNaming` service provider that uses the mechanisms defined in the Interoperable Name Service specification to contact the server's `CosNaming` service and to resolve the `EJBHome` object using standard `CosNaming` APIs. The JNDI `CosNaming` service provider may or may not use the JNDI SPI architecture. The JNDI `CosNaming` service provider must access the root `NamingContext` of the server's `CosNaming` service by creating an object reference from the following URL:

`corbaloc:iiop:1.2@<host>:<port>/<objectkey>` (where `<host>`, `<port>`, and `<objectkey>` are the values corresponding to the root `NamingContext` advertised by the server's `CosNaming` Service), or by using an equivalent mechanism.

At deployment time, the developer of the client container should obtain the host, port and object key of the server's `CosNaming` service and the `CosNaming` name of the server `EJBHome` object (for example, by browsing the server's namespace) for each `ejb-ref` element in the client components's deployment descriptor. The `ejb-ref-name` (which is used by the client code in the JNDI lookup call) should then be linked to the `EJBHome` object's `CosNaming` name. At runtime, the client component's JNDI lookup call uses the `CosNaming` service provider, which contacts the server's `CosNaming` service, resolves the `CosNaming` name, and returns the `EJBHome` object reference to the client component.

Since the `EJBHome` object's name is scoped within the namespace of the `CosNaming` service that is accessible at the provided host and port, it is not necessary to federate the namespaces of the client and server containers.

The advantage of using `CosNaming` is better integration with the IIOP infrastructure that is already required for inter operability, as well as inter operability with non-J2EE CORBA clients and servers. Since `CosNaming` stores only CORBA objects it is likely that vendors will use other enterprise directory services for storing other resources.

Sun Java System Application Server incorporates all the naming resources of JNDI, based on the J2EE 1.3 specification.

**CosNaming provider** To support a global JNDI namespace (accessible to IIOP application clients), Sun Java System Application Server includes J2EE based CosNaming provider which supports binding of CORBA references (remote EJB references). The `InitialContext` returned to the IIOP clients is a CosNaming provider. An instance of Sun Java System Application Server server registers the entity beans for the IIOP clients to lookup and bind to.

Note that Sun Java System Application Server treats objects stored in CosNaming and the local JNDI naming environment are transient: that is, on each server startup as well as application reloading, all relevant objects are rebound to the namespace again. To know more about configuring support for CORBA/IIOP clients, see [“Configuring the Server For Corba/IIOP Clients” on page 303.](#)

## JNDI Connection Factories

For J2EE web applications, the deployment descriptor in the `web.xml` file is the placeholder for defining references to application environment entries, resource manager (such as SQL Data Source) connection factories, or EJBs. Applications look up such references using the JNDI `InitialNamingContext` provided by the J2EE containers. This makes applications portable to different Sun Java System application server environments by just making changes to the deployment descriptor, that is, without accessing or modifying the application's source code. Likewise, J2EE requires the deployment descriptors for entity beans (`ejb-jar.xml`) as well as the IIOP application clients (`application-client.xml`) to be the primary vehicles for these JNDI naming references.

A connection factory is an object that produces connection objects that enable a J2EE component to access a resource. The connection factory for a database is a `javax.sql.DataSource` object, which creates a `java.sql.Connection` object.

In Sun Java System Application Server, you can configure the means of accessing the following resources and resource factories:

- JDBC connection factories
- JMS Connection factories based on MQ
- JavaMail Session connection factories
- JCA Connector factories

- Generic, custom user-written resource object factories.
- Support for external resource repositories such as LDAP

All Sun Java System Application Server resource factories are specified within the `<resources>` `</resources>` tags in `server.xml` and have a JNDI name specified using the `jndi-name` attribute. This attribute is used to register the factory in the server-wide namespace. Deployers can map user-specified, application-specific resource reference names (declared within `resource-ref` or `resource-env-ref` elements) to these server-wide resource factories using the `resource-ref-mapping` element. This enables deployment time decisions to be made with regards to which JDBC drivers (and other resource factories) to use for a given application.

A custom resource accesses a local JNDI repository and an external resource accesses an external JNDI repository. Both types of resources need user-specified factory class elements, JNDI name attributes, etc. In this section, we will discuss how to configure JNDI connection factory resources, for J2EE resources, and how to access these resources.

The following topics are covered in this section:

- [To Create a Custom Resource](#)
- [To Create an External JNDI Resource](#)
- [Accessing External JNDI Repositories](#)
- [Mapping Application Resource References](#)
- [About URL Connection Factory Resources](#)
- [Mapping Application Resource Environment References](#)
- [Mapping EJB References](#)

## To Create a Custom Resource

The `custom-resource` element defined in `server.xml` provides a way of specifying a custom server-wide resource object factory. Such object factories implement the `javax.naming.spi.ObjectFactory` interface. This element associates a JNDI name (specified through the `jndi-name` sub-element like other Sun Java System Application Server resources) to be used in the server-wide namespace, its type, name of the resource factory class and a set of standard properties used to instantiate the same.

The following example illustrates the implementation of the `javax.naming.spi.ObjectFactory` interface:

```
<resources> <custom-resource jndi-name="test/myBean"
res-type="test.MyBean" factory-class="test.MyBeanFactory"
enabled="true">

<property name="foo" value="test custom bean prop" />
</custom-resource>
</resources>
```

You need to ensure that the resource reference's environment references and EJB references are linked to the configured server-wide resources defined using the `custom-resource` and `external-jndi-resource` tags in `server.xml`. Dynamic redeployment of application components is an issue for the JNDI naming environment. Sun Java System Application Server will release all the application specific references and rebind all the new references into the newly installed application's naming context.

To create a custom resource using the Administration interface:

1. In the left pane of the Administration interface, open the Sun Java System Application Server instance whose JNDI configuration you want to modify.
2. Open the JNDI tab and click Custom Resources. If any custom resources have been created already, they will be listed in the right pane. To create a new custom resource, click New. Open the JNDI tab and click Custom Resources. If any custom resources have been created already, they will be listed in the right pane. To create a new custom resource, click New. A page for adding a new custom resource appears.
3. In the JNDI Name field, enter the name to use to access the resource. This name will be registered in the JNDI naming service.
4. In the Resource Type field, enter a fully qualified type definition, as shown in the example above. Your Resource Type definition should follow this format:  
xxx.xxx.
5. In the Factory Class field, enter a factory class name for the custom resource you are creating. The Factory Class is the user-specified name for the factory class. This class implements the `javax.naming.spi.ObjectFactory` interface.
6. In the Description field, enter a description for the resource you're creating. This description is a string value and can comprise a maximum of 250 characters.
7. Mark the Custom Resource Enabled checkbox, to enable the custom resource.
8. Click OK to save your custom resource.



## To Create an External JNDI Resource

To create an external resource using the Administration interface:

1. In the left pane of the Administration interface, open the Sun Java System Application Server instance whose JNDI configuration you want to modify.
2. Open JNDI and select External Resources. If any external resources have been created already, they will be listed in the right pane. To create a new external resource, click New.
3. In the JNDI Name field, enter the name that is to be used to access the resource. This name will be registered in the JNDI naming service.
4. In the Resource Type field, enter a fully qualified type definition, as shown in the example above. Your Resource Type definition should follow this format:  
`xxx.xxx`.
5. In the JNDI Lookup field, enter the JNDI value to look up in the external repository. For example, if you are creating an external resource to connect to an external repository, to test a bean class, your JNDI Lookup could read  
`cn=testmybean`.
6. In the Factory Class field, enter a JNDI factory class external repository, for example, `com.sun.jndi.ldap`. This class implements the `javax.naming.spi.ObjectFactory` interface.
7. In the Description field, enter a description for the resource you're creating. This description is a string value and can comprise a maximum of 250 characters.
8. Mark the External Resource Enabled checkbox, to enable the external resource.
9. Click OK to save your custom resource.

## Accessing External JNDI Repositories

Often applications running on Sun Java System Application Server require access to resources stored in an external JNDI repository. For example, generic Java objects could be stored in an LDAP server as per the Java schema. External JNDI resource elements let users configure such external resource repositories. The external JNDI factory must implement `javax.naming.spi.InitialContextFactory` interface.

**Example:**

```

<resources>
<!-- external-jndi-resource element specifies how to access J2EE resources
-- stored in an external JNDI repository. The following example
-- illustrates how to access a java object stored in LDAP.
-- factory-class element specifies the JNDI InitialContext factory that
-- needs to be used to access the resource factory. property element
-- corresponds to the environment applicable to the external JNDI context
-- and jndi-lookup-name refers to the JNDI name to lookup to fetch the
-- designated (in this case the java) object.
-->
<external-jndi-resource jndi-name="test/myBean"
jndi-lookup-name="cn=myBean"
res-type="test.myBean"
factory-class="com.sun.jndi.ldap.LdapCtxFactory">

<property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
<property name="SECURITY_AUTHENTICATION" value="simple" />
<property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
<property name="SECURITY_CREDENTIALS" value="changeit" />
</external-jndi-resource>
</resources>

```

**Mapping Application Resource References**

Application-specific resource references must be mapped to pre-defined server-wide resource factories. The Sun Java System Application Server specific resource reference mapping element is used for this.

In the following example, we look at a web application's deployment descriptor `web.xml` where a resource reference is specified to a JDBC `DataSource`.

```

<resource-ref>
<res-ref-name> jdbc/EstoreDataSource </res-ref-name>
<res-type> javax.sql.DataSource </res-type>
<res-auth> Container </res-auth>
</resource-ref>

```

The desired `res-ref-name` can also be mapped to the container-wide Oracle JDBC connection resource factory, as follows:

```

<resource-ref>
<res-ref-name> jdbc/EstoreDataSource </resource-ref-name>
<jndi-name> jdbc/estore/InventoryDB </jndi-name>
</resource-ref>

```

## About URL Connection Factory Resources

URL connection factories do not require any resource to be defined in `server.xml`. The `jndi-name` element of the corresponding Sun Java System Application Server application (web or ejb) deployment descriptor specifies the target URL.

For example, let us assume that a web application's deployment descriptor `web.xml` specifies a `java.net.URL` resource reference and this is mapped to the URL <http://www.sun.com/index.html> in `sun-web.xml`:

The mapping would be as follows:

```
<resource-ref>
<res-ref-name>myURL</res-ref-name>
<res-type>java.net.URL</res-type>
<res-auth>Container</res-auth>
</resource-ref>

<sun-web-app>
<resource-ref>
<res-ref-name>myURL</res-ref-name>
<jndi-name> http://www.sun.com/index.html </jndi-name>
</resource-ref>
</sun-web-app>
```

## Mapping Application Resource Environment References

Application-specific resource environment reference declarations must be mapped to target resource objects available in the application server's runtime environment. The resource environment mapping element defined in the Sun Java System Application Server-specific configuration file lets deployers map as follows:

Example:

```
<resource-env-ref>
<description> My Topic </description>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<res-env-ref-type> javax.jms.Topic </res-type>
</resource-env-ref>
```

This reference is mapped to the `jms/IMQ/Topics/Stocks/SUNW` topic defined in `server.xml`. See the *Sun Java System Application Server Administrator's Configuration File Reference* for more information.

```
<resource-env-ref-mapping>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<jndi-name> jms/IMQ/Topics/Stocks/SUNW </jndi-name>
</resource-env-ref-mapping>
```

## Mapping EJB References

It's also possible to decouple the actual `ejb-name` used in the application code from the `ejb-name` used for the target enterprise bean. This is particularly useful when you do not want to modify the web application deployment descriptor, `web.xml` and use the `ejb-name` of the enterprise bean deployment descriptor. The Sun Java System Application Server specific configuration allows you to map the `ejb-ref-name` element to the target bean's `ejb-name` without using the `ejb-ref-mapping` element in the Sun Java System Application Server specific deployment descriptor.

Example:

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<ejb-ref-type> Entity </ejb-ref-type>
<home> com.wombat.empl.EmployeeRecordHome </home>
<remote> com.wombat.empl.EmployeeRecord </remote>
</ejb-ref>

<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<jndi-name> AccountEJB </jndi-name>
</ejb-ref-mapping>
```

## About Persistence Manager Resources

This module describes persistence and establishes a framework for the use of the pluggable Persistence Managers that are supported by Sun Java System Application Server.

This module covers the following topics:

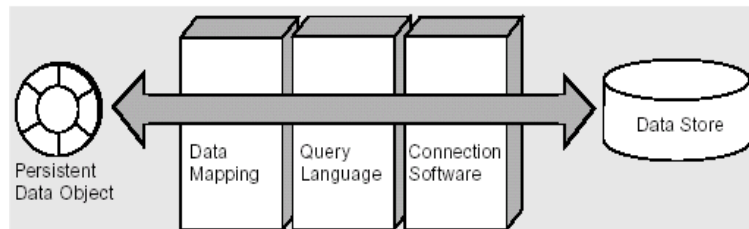
- [What is Persistence?](#)
- [Role of the Persistence Manager](#)
- [Pre-Deployment Bean Configuration](#)
- [Creating a New Persistence Manager](#)

## What is Persistence?

A key aspect of most business applications is the programmatic manipulation of persistent data; long-lived data stored outside of an application. Although persistent data is read into transient memory for the purpose of using or modifying it, it is written out to a relational database or flat file system for long-term storage.

In object-oriented programming systems, persistent data is represented in memory as one or more data objects manipulated by application code. In general, the correspondence between persistent data in a data store and its representation as a persistent data object in memory is achieved through a number of software layers as shown in the following figure, [Figure 9-2](#):

**Figure 9-2** Basic Persistence Scheme



Each data store has an interface to the outside world through driver software used to set up and maintain a connection between the data store and an application. With this connection established, a query language is used to retrieve information from the data store and read it into an application, or conversely, to write data from the application into the data store. Another layer provides a mapping between data objects in memory and the information in the data store.

Through this general scheme, programmers can represent persistent data as runtime objects to be used and manipulated by an application. The scheme supports all basic persistence operations—often abbreviated as CRUD:

- C - Creating persistent data (inserting in a data store)
- R - Retrieving persistent data (selecting from a data store)
- U - Updating persistent data
- D - Deleting persistent data

## Role of the Persistence Manager

The Persistence Manager (PM) is responsible for the persistence of entity beans with Container-managed Persistence, in the EJB container. The entity bean provider is responsible for providing the entity bean class as an abstract class. The Persistence Manager provider's tools are responsible for providing the concrete implementations. They can achieve this by sub-classing the abstract entity bean and related classes and providing an concrete implementation or by utilizing encapsulation and delegation.

The classes provided by the persistence manager's tools are responsible for managing the relationships between the entity beans and for managing the access to their persistent state. The PM tools are also responsible for providing the implementations of the `java.util.Collection` classes that are used in the maintaining the container managed relationships (CMRs).

## Pre-Deployment Bean Configuration

The Enterprise Java Beans standard provides two types of persistence for Entity Beans. These are the Container Managed Persistence (CMP) and Bean Managed Persistence (BMP) The EJB 2.0 specification does not define a standard API between a EJB server and the persistence manager.

---

<b>NOTE</b>	For CMP, relationships should always be mapped to the Primary Key field(s) of the related table.
-------------	--

---

This section describes the integration requirements at deployment and code-generation. Deployment can be used to mean multiple things. Generally speaking, the deployment process can be thought of three distinct steps: Configuration, Code-generation and installation.

A number of properties must be specified for a bean, including the persistence mechanism used, persistence vendor, and the version in use, and additional information required by the persistence mechanism. Most of the persistence vendors have a concept of a project, which represents all the related beans and their dependant classes, which can be deployed as a single unit. There can be a vendor specific xml file per project.

The three standard files supported for deployment purposes include `ejb-jar.xml`, `sun-ejb-jar.xml` and `sun-cmp-mappings.xml`. Each EJB module with CMP beans in the `sun-ejb-jar.xml` must have a `<pm-descriptors>` with at least one `<pm-descriptor>` element, which specifies five more attributes. These five attributes are `pm-identifier`, `pm-version`, `pm-config`, `pm-class-generator`, and `pm-mapping-factory`.

The Sun Java System Application Server-specific descriptor (as in `sunEjb_jar_2_0.DTD`) defines the persistence manager related tags. A sample CMP descriptor might look like the one below as defined in the Sun Java System Application Server DTD:

PM descriptors contain one or more pm descriptors, but only of them must be in use at any given time

```
-->

<!--
pm-descriptor describes the properties for the persistence manager
associated with entity bean
-->

<!--
pm-descriptor ( pm-identifier, pm-version, pm-config?,
pm-class-generator?,
pm-mapping-factory?)>

<!--
```

This element describes the vendor who provided the PM implementation, for example this could be Sun Java System Application Server Transparent Persistence, TopLink, Versant or CoboBase:

```
-->

<!--
pm-identifier (#PCDATA)>

<!--
pm-version further specifies which version of PM vendor product to be used
-->

<!--
pm-version (#PCDATA)>

<!--
pm-config specifies the vendor specific config file to be used
-->
```

```

<!ELEMENT pm-config (#PCDATA)>

<!--
pm-class-generator specifies the vendor specific concrete class generator
This is the name of the class specific to a vendor:
-->

<!ELEMENT pm-class-generator (#PCDATA)>

<!--
pm-mapping-factory specifies the vendor specific mapping factory
This is the name of the class specific to a vendor:
-->

<!ELEMENT pm-mapping-factory (#PCDATA)>

```

## Creating a New Persistence Manager

Using the Administration interface, you can create a new Persistence Manager instance. To create a new persistence manager instance:

1. From the left pane of the Administration interface, open the Sun Java System Application Server instance for which you want to create a new Persistence Manager. Click Persistence Manager from the list of server components displayed.  
  
If any persistence managers have been created for that specific instance of Sun Java System Application Server, you will see the list displayed in the right pane of the Administration interface.
2. To create a new Persistence Manager, click New. A page for creating a new persistence manager appears.
3. Enter a JNDI name. This is the JNDI Name used by the application server run-time to locate a particular Persistence Manager on behalf of an application. The name must be the same as that defined in the entity bean's cmp-resource element of the sun specific deployment descriptor.
4. In the Description field, provide a description for the new persistence manager. The value for this field is string, and can comprise up to 250 characters.



5. In the Factory Class field, provide a factory class connection for the persistence manager. The `setEntityContext` looks up this connection factory through the JNDI name lookup. The factory class name is the classname of the Persistence Manager Factory that creates Persistence Manager instances. By default this will be set to the Sun Java System Application Server's internal Persistence Manager Factory class; if you use an alternative implementation you must ensure that this class is available in the server classpath.
6. From the Connection Pool drop-down list, select a database connection pool, into which the new persistence manager will be pooled. With connection pooling, an entity bean can request a single connection and use it to execute concurrent statements for multiple client threads. Just like any other database access, the Persistence Manager will use connection pooling to improve performance and scalability. Choose an existing connection pool or "None Selected" if you haven't yet created the pool.

Note: a JDBC Resource will be automatically created to allow the PM runtime to bind to the connection pool using JNDI - the JNDI name of the JDBC Resource will be the same as the PM JNDI Name with a prefix of "PM." Deleting a Persistence Manager will also delete the associated JDBC Resource.

7. To enable the persistence manager, mark the Persistence Manager Enabled checkbox. The persistence manager is now enabled for the connection factory specified.
8. Click OK to save your changes.

## About JDBC Resources

This module talks about the JDBC API in general, and JDBC resources and their implementation and usage in Sun Java System Application Server, in specific.

This module consists of the following sections:

- [About the JDBC API](#)
- [About Database Access Models](#)
- [About JDBC Datasources](#)
- [About JDBC Connections](#)
- [About JDBC Transactions](#)

## About the JDBC API

The JDBC API is a Java API for accessing virtually any kind of tabular data. (As a point of interest, JDBC is the trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for “Java Database Connectivity.”) The JDBC API consists of a set of classes and interfaces written in the Java programming language that provide a standard API for tool/database developers and makes it possible to write database applications using an all-Java API.

The JDBC API makes it easy to send SQL statements to relational database systems and supports all dialects of SQL. But the JDBC 3.0 API goes beyond SQL, also making it possible to interact with other kinds of data sources, such as files that are outside of a database.

The value of the JDBC API is that an application can access virtually any data source and run on any platform with a Java virtual machine. In other words, with the JDBC API, it isn't necessary to write one program to access a Sybase database, another program to access an Oracle database, another program to access an IBM DB2 database, and so on. You can write a single program using the JDBC API, and the program will be able to send SQL or other statements to the appropriate data source. And, with an application written in the Java programming language, you do not have to worry about writing different applications to run on different platforms. The combination of the Java platform and the JDBC API lets a programmer write once, and run the code from anywhere.

### What Does The JDBC API Do?

A JDBC technology-based driver (JDBC driver) makes it possible to do three things:

- Establish a connection with a data source
- Send queries and update statements to the data source
- Process the results

The following code fragment gives a simple example of these three steps:

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/AcmeDB");
Connection con = ds.getConnection("myLogin", "myPassword");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT a,b,c FROM Table1");
```

```

while (rs.next()) {
    int x = rs.getInt("a");
    String s = rs.getString("b");
    float f = rs.getFloat("c");
}

```

## About Database Access Models

The JDBC API supports both two-tier and three-tier models for database access. Sun Java System Application Server incorporates the more popular, two-tier database access model.

This section covers the following topics:

- [Two-Tier Database Access Model](#)
- [Three-Tier Database Access Model](#)

### *Two-Tier Database Access Model*

In the two-tier database access model a Java applet or application talks directly to the data source using a DBMS-proprietary protocol. This access model requires a JDBC driver that can communicate with the particular data source being accessed. A user's commands are delivered to the database or other data source, and the results of those statements are sent back to the user. The data source may be located on another machine to which the user is connected via a network. This configuration is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server. The network can be an intranet, which, for example, connects employees within a corporation, or it can be the Internet.

### *Three-Tier Database Access Model*

In the three-tier database access model, the Java applet or application sends commands to a “middle tier” of services, which then sends the commands to the data source. The client application communicates with the middle tier via HTTP, RM, CORBA, or other calls. The middle tier communicates with the data store through a DBMS-proprietary protocol. The data source processes the commands and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier

makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that it simplifies the deployment of applications. Finally, in many cases, the three-tier architecture can provide performance advantages.

## About JDBC Datasources

A `DataSource` object is the representation of a data source in the Java programming language. In basic terms, a data source is a facility for storing data. It can be as sophisticated as a complex database for a large corporation or as simple as a file with rows and columns. A data source can reside on a remote server, or it can be on a local desktop machine. Applications access a data source using a connection, and a `DataSource` object can be thought of as a factory for connections to the particular data source that the `DataSource` instance represents. The `DataSource` interface provides two methods for establishing a connection with a data source.

A `DataSource` object has properties that identify and describe the data source it represents. Also, a `DataSource` object works with a JNDI naming service and is created, deployed, and managed separately from the applications that use it. A driver vendor will provide a class that is a basic implementation of the `DataSource` interface as part of its JDBC 2.0 or 3.0 driver product.

This section covers the following topics:

- [Properties Of a DataSource Object](#)
- [Registering a JDBC Resource](#)

### Properties Of a DataSource Object

A `DataSource` object has a set of properties that identify and describe the real world data source that it represents. These properties include information like the location of the database server, the name of the database, the network protocol to use to communicate with the server, and so on. `DataSource` properties follow the `JavaBeans` design pattern and are usually set when a `DataSource` object is deployed.

To encourage uniformity among `DataSource` implementations from different vendors, the JDBC 2.0 API specifies a standard set of properties and a standard name for each property.

An instance of a class that implements the `DataSource` interface represents one particular data source. Every connection produced by that instance will reference the same data source. In a basic `DataSource` implementation, a call to the method `DataSource.getConnection` returns a connection object that, like the connection object returned by the `DriverManager` facility, is a physical connection to the data source.

JNDI provides a uniform way for an application to find and access remote services over the network. The remote service may be any enterprise service, including a messaging service or an application-specific service, but, of course, a JDBC application is interested mainly in a database service. Once a `DataSource` object is created and registered with a JNDI naming service, an application can use the JNDI API to access that `DataSource` object, which can then be used to connect to the data source it represents.

`DataSource` objects that implement connection pooling likewise produce a connection to the particular data source that the `DataSource` class represents. The connection object that the method `DataSource.getConnection` returns, however, is a handle to a `PooledConnection` object rather than being a physical connection. An application uses the connection object just as it usually does and is generally unaware that it is in any way different. Connection pooling has no effect whatever on application code except that a pooled connection, as is true with all connections, should always be explicitly closed. When an application closes a connection that is pooled, the connection joins a pool of reusable connections. The next time `DataSource.getConnection` is called, a handle to one of these pooled connections will be returned if one is available. Because connection pooling avoids creating a new physical connection every time one is requested, it can help to make applications run significantly faster.

A `DataSource` class can likewise be implemented to work with a distributed transaction environment. An EJB server, for example, supports distributed transactions and requires a `DataSource` class that is implemented to interact with it. In this case, the `DataSource.getConnection` method returns a `Connection` object that can be used in a distributed transaction. As a rule, EJB servers provide support for connection pooling as well as distributed transactions. Like connection pooling, transaction management is handled internally, so using distributed transactions is easy. The only requirement is that when a transaction is distributed (this involves two or more data sources), the application cannot call the transaction methods as either `commit` or `rollback`. It also cannot put the connection in `auto-commit` mode. The reason for these restrictions is that a transaction manager begins and ends a distributed transaction under the covers, so an application cannot do anything that would affect the time that a transaction begins or ends. To know more about Java transactions, please see [Chapter 8, “Using Transaction Services.”](#)

## Registering a JDBC Resource

You can register a JDBC resource with Sun Java System Application Server using either the Administration interface or the command-line interface.

This section covers the following topics:

- [Registering a Resource Using the Command Line](#)
- [Registering a Resource Using the Administration Interface](#)

### *Registering a Resource Using the Command Line*

To register a JDBC resource using the command line interface, run the following command:

```
./asadmin create-jdbc-resource
```

The XML snippet, for registering a JDBC resource, would have to specify a few attributes, as below (excerpted from `sun-server_7_0.dtd`).

```
<!-- JDBC javax.sql.DataSource resource definition -->
<!ELEMENT jdbc-resource (description?, property*)>
<!-- ATTLIST jdbc-resource jndi-name CDATA #REQUIRED
pool-name CDATA #REQUIRED
enabled %boolean; 'true' -->
```

Note that all this specifies, is the symbolic name with which applications will refer to this data source, from inside J2EE applications. `pool-name` attribute points to a named pool definition, that specifies all aspects of database connectivity. The `enabled` attribute can be used by the administrator to turn off some resources.

### *Registering a Resource Using the Administration Interface*

To register a datasource using the Administration interface:

1. In the left pane of the Administration interface, open the Sun Java System Application Server instance for which you want to register a JDBC resource.
2. Open JDBC.
3. Under JDBC, click JDBC Resource.
4. In the right pane, click New. The page for creating a new JDBC resource, appears in the right pane.

5. Provide the JNDI name for the resource that you are creating.

JDBC Resources are stored in the JNDI repository and are accessed using the JNDI name. The JNDI name has an explicit root at `Java:comp:env/` so you do not need to specify that part of the name. It is recommended that JDBC Resources (DataSources) are stored under the 'jdbc' sub-context so your JNDI name would be similar to `jdbc/EmployeeDB_DS`.

6. Select a pool name for the new datasource, from the list of pool names in the Pool Name drop-down list. All registered connection pools will appear in this drop-down list. The pool name that you select points to a named pool definition, that specifies all aspects of database connectivity. More than one JDBC resource can use a single pool definition. To know more about how to configure JDBC connection pools, see [“Creating a New JDBC Connection Pool Using the Administration Interface” on page 250](#).
7. In the Description field, provide a brief description that describes the purpose of the data source. Your description must not exceed 250 characters.
8. Mark the Enabled checkbox to enable or disable a datasource. A datasource cannot be used to connect to a database unless it is enabled.
9. Click OK to register the new datasource, or click Cancel to cancel the new datasource. When you click cancel, you are returned to the main JDBC resources page, from where you can create a new datasource again.

## About JDBC Connections

A Connection object represents a connection with a database. A connection session includes the SQL statements that are executed and the results that are returned over that connection. A single application can have one or more connections with a single database, or it can have connections with many different databases.

A user can get information about a Connection object's database by invoking the `Connection.getMetaData` method. This method returns a `DatabaseMetaData` object that contains information about the database's tables, the SQL grammar it supports, its stored procedures, the capabilities of this connection, and so on.

An application uses a Connection object produced by a DataSource object. As is always the case, an application should include a “finally” block to assure that connections are closed even if an exception is thrown. This is even more important if the Connection object is a pooled connection because it makes sure that a valid connection will always be put back into the pool of available connections. The following code fragment, in which `con` is Connection object, is an example of a finally block that closes a connection if it is valid.

```
finally{
    if (con != null) con.close();
}
```

Note that a `finally` block appears after a `try/catch` block, as shown in the following example, where `ds` is a `DataSource` object.

```
try {
    Connection con = ds.getConnection("user", "secret");
    // . . . code to do the application's work
} catch {
    // . . . code to handle an SQLException
} finally {
    if (con != null) con.close();
}
```

This section covers the following topics:

- [About JDBC URLs](#)
- [Configuring JDBC Connection Pools](#)
- [About Connection Pooling](#)
- [Monitoring JDBC Connection Pooling](#)
- [About Connection Sharing](#)

## About JDBC URLs

A URL (Uniform Resource Locator) gives information for locating a resource on the Internet. It can be thought of as an address.

A JDBC URL provides a way of identifying a data source so that the appropriate driver will recognize it and establish a connection with it. Driver writers are the ones who actually determine what the JDBC URL that identifies a particular driver will be. Users do not need to worry about how to form a JDBC URL; they simply use the URL supplied with the drivers they are using. JDBC's role is to recommend some conventions for driver writers to follow in structuring their JDBC URLs.

Since JDBC URLs are used with various kinds of drivers, the conventions are, of necessity, very flexible. First, they allow different drivers to use different schemes for naming databases. The ODBC sub-protocol, for example, lets the URL contain attribute values (but does not require them).



Second, JDBC URLs allow driver writers to encode all necessary connection information within them. This makes it possible, for example, for an applet that wants to talk to a given database to open the database connection without requiring the user to do any system administration chores.

Third, JDBC URLs allow a level of indirection. This means that the JDBC URL may refer to a logical host or database name that is dynamically translated to the actual name by a network naming system. This allows system administrators to avoid specifying particular hosts as part of the JDBC name. There are a number of different network name services, and there is no restriction about which ones can be used.

The standard syntax for JDBC URLs is shown here. It has three parts, which are separated by colons.

`jdbc:<subprotocol>:<subname>`

The three parts of a JDBC URL are broken down as follows:

- `jdbc`

The protocol in a JDBC URL is always `jdbc`.

- `<subprotocol>`

The name of the driver or the name of a database connectivity mechanism, which may be supported by one or more drivers. A prominent example of a sub-protocol name is ODBC, which has been reserved for URLs that specify ODBC-style data source names. For example, to access a database through a JDBC-ODBC bridge, one might use a URL such as `jdbc:odbc:fred`.

In this example, the subprotocol is ODBC, and the subname `fred` is a local ODBC data source.

If one wants to use a network name service (so that the database name in the JDBC URL does not have to be its actual name), the naming service can be the subprotocol. So, for example, one might have a URL such as:

`jdbc:dcenaming:accounts-payable`

In this example, the URL specifies that the local DCE naming service should resolve the database name `accounts-payable` into a more specific name that can be used to connect to the real database.

- `<subname>:`

A way to identify the data source. The subname can vary, depending on the subprotocol, and it can have any internal syntax the driver writer chooses, including a `sub-subname`. The point of a `subname` is to give enough information to locate the data source. In the previous example, `fred` is enough because ODBC provides the remainder of the information. A data source on a remote server requires more information, however. If the data source is to be accessed over the Internet, for example, the network address should be included in the JDBC URL as part of the `subname` and should adhere to the following standard URL naming convention:

*//hostname:port/subsubname*

Supposing that `dbnet` is a protocol for connecting to a host on the Internet, a JDBC URL might look like this:

`jdbc:dbnet://wombat:356/fred`

## Configuring JDBC Connection Pools

Sun Java System Application Server allows users to create named JDBC connection pools. A JDBC connection pool defines the properties used to create a connection pool. A pool definition is named, and a definition can be reused to configure multiple JDBC resources. Each named pool definition results in an instantiation of a physical pool at server start-up. If two or more JDBC resources point to the same pool definition, they will be using the same pool of connections at run time.

You can create and configure JDBC connection pools using the Administration interface and the Command Line Interface, as detailed in the following sections:

- [Creating a New JDBC Connection Pool Using the Administration Interface](#)
- [Creating a New JDBC Connection Pool Using the Command-Line Interface](#)
- [Using the Command Line Interface to Manage JDBC Connection Pools](#)

### *Creating a New JDBC Connection Pool Using the Administration Interface*

To create a new JDBC connection pool using the Administration interface, perform the following tasks:

1. In the left pane of the Administration interface, open the Sun Java System Application Server instance for which you want to create a JDBC connection pool.
2. Select JDBC from the list of J2EE services listed under your Sun Java System Application Server, and open the `ConnectionPools` tab under it. The page for creating a new JDBC connection pool appears in the right pane.

3. In the Name field, provide a JNDI name of the connection pool that you are creating.
4. Mark the Global Transaction Support Enabled checkbox to enable global Transaction support for the new connection pool. Connection pools that are capable of participating in global transactions, and are referred to as XA-capable connection pools.
5. Select a database vendor, from the Database Vendor drop-down list, and click Next. You need to configure connection pool settings in the screen that is displayed next.

### *Configuring Connection Pool Settings*

To configure pool settings, perform [Step 1](#) to [Step 5](#) in “[Creating a New JDBC Connection Pool Using the Administration Interface](#)” on [page 250](#). When you click Next, as described in [Step 5](#), a new page appears in the right pane of the Administration interface. It contains the following sections:

- General
- Properties
- Pool Settings
- Connection Validation
- Transaction Isolation

In the General section of this page, specify the values for the parameters provided, based on the guidelines given in the following table:

**Table 9-2** General Settings

Parameter	Description
Name	The name of the connection pool.
DataSource ClassName	The vendor-specific classname that implements the DataSource and / or XADataSource APIs.
Description	The description of the connection pool.

In the Properties section of this page you specify standard and proprietary JDBC connection pool properties; many of these properties are optional. By default the names of all of the standard properties are provided. You will need to consult your database vendor's documentation to determine which standard and vendor specific properties are required.

In the Pool Settings section of this window, specify the values for the parameters provided, based on the guidelines given in the following table:

**Table 9-3**     Connection Pool Settings

Parameter	Description
Steady Pool Size	Specify the minimum number of connections that must be maintained in the pool. When a connection is given to a requesting thread, it is removed from the pool, reducing the current pool size. The steady pool size also refers to the number of entries that will be added to the pool on server startup.
Max Pool Size	Specify the maximum number of connections that can be allowed in the pool at any given point in time.
Pool Resize Quantity	When the pool shrinks toward the steady pool size it is resized in batches. This value determines the size of the batch. Making this value too large will delay connection recycling, making it too small will be less efficient. Note, the pool capacity is only ever increased one connection at a time so this field does not effect increases in pool capacity.
Idle Timeout (secs)	The maximum time in seconds that a connection can remain idle in the pool. After this time, the pool implementation can close this connection.
Max Wait tim	The amount of time the caller will wait before getting a connection timeout. The default wait time is long, which means that a caller can wait for a long time.

In the Connection Validation and Transaction Isolation sections of this window, select the validation method and transaction isolation methods for the connection pool, based on the guidelines given in the following table:

**Table 9-4**     Connection Validation and Transaction Isolation

Parameter	Description
Connection Validation Required	If this field is checked then connections will be validated before they are passed to the application. This allows the application server to automatically re-establish database connections in the case of the database becoming unavailable due to network failure or database server crash. Validation of connections will incur additional overhead and slightly reduce performance.

**Table 9-4** Connection Validation and Transaction Isolation

Parameter	Description
Validation Method	<p>There are three methods that the application server can employ to validate database connections; you need to understand the capabilities of your database to determine the appropriate one. The three validation methods are:</p> <ul style="list-style-type: none"> <li>• <code>auto-commit</code>, <code>meta-data</code> - the <code>con.getAutoCommit()</code> and <code>con.getMetaData()</code> methods are commonly used to validate a connection, unfortunately many JDBC drivers cache the results of these calls so they do not always provide a reliable validation. You should check with your vendor to determine whether these calls are cached or not.</li> <li>• <code>table</code>: this method requires the app server to perform a query on a user specified table. The actual query is "select (count *) from &lt;table-name&gt;". The table must exist and be accessible, though it doesn't require any rows. You should not use an existing table that has a large number of rows or a table that is already frequently accessed.</li> </ul>
Table Name	If you select the last validation option, <code>table</code> , from the Validation Method drop-down list, specify the table name here.
Fail All Connections	Check this box to fail all connections in the pool and re-establish them if a single connection is determined to have failed. If left unchecked, connections will be individually re-established only when they are used.
Transaction Isolation	Allows you to select the transaction isolation level for this connection. If left unspecified the pool operates with default isolation level provided by the JDBC Driver.
Guarantee Isolation Level	Only applicable if the isolation level has been specified. This ensures that any connection taken from the pool will have the same isolation level. For example if the isolation level for the connection was changed programatically (for example, <code>con.setTransactionIsolation</code> ) when last used this mechanism will change it back to the specified isolation level.

### *Creating a New JDBC Connection Pool Using the Command-Line Interface*

This section describes, through the use of examples, how to create JDBC connection pools using the Command Line Interface.

The following table lists all the options that you need to create connection pools, such as server name, password. Sample values have been used in the following table. It is recommended that you keep the parameters specific to your installation of Sun Java System Application Server ready, before running the commands explained in this section.

**Table 9-5** Options required for creating JDBC Connection Pools using the Command Line Interface

Description of the required option	Sample Value
Sun Java System Application Server admin user name	<i>admin</i>
Sun Java System Application Server admin password	<i>adminadmin</i>
Sun Java System Application Server admin port	<i>8888</i>
Sun Java System Application Server machine name	<i>sas.sun.com</i>
Sun Java System Application Server instance name	<i>server1</i>
Datasource classname for the connection pool	<i>oracle.jdbc.xa.client.OracleXADataSource</i>  Note: Use the datasource classname of the database for which you are creating the connection pool. The database used in this example is Oracle.
Jdbc resource description Sample	<i>Jdbc Resource</i>
Connection pool description Sample	<i>Jdbc Connection Pool</i>
Jdbc resource name	<i>jdbc/SampleJdbcResource</i>
Connection pool name	<i>SampleJdbcConnectionPool</i>
Database user name	<i>oracle</i>
Database password	<i>oracle</i>
Jdbc connection URL	<i>jdbc:oracle:thin:@oracleserver.sun.com:1521:ORA</i>

The following example makes use of the variables listed in [Table 9-5](#).

**Example 1:**

This example creates a JDBC connection pool called `SampleJdbcConnectionPool`. A two-step process is used, in this example, to create the JDBC connection pool, as follows:

- [Step 1 - Create a Connection Pool](#)
- [Step 2 - Apply Changes To the Instance](#)

### Step 1 - Create a Connection Pool

The following is the command line interface syntax for creating a JDBC connection pool:

```
asadmin create-jdbc-connection-pool --user admin_user [--password
admin_password] [--host localhost] [--port 4848] [--secure | -s] [--instance
instancename] --datasourceclassname classname [--restype res_type]
[--steadypoolsize 8] [--maxpoolsize 32] [--maxwait 60000] [--poolresize 2]
[--idletimeout 300] [--isolationlevel isolation_level] [--isolationguaranteed]
[--isconnectvalidatereq=false] [--validationmethod auto-commit]
[--validationtable tablename] [--failconnection=false] [--description text]
[--property (name=value)[:name=value]*] connectionpool_id
```

For example, the following command will create a connection pool called `SampleJdbcConnectionPool`.

```
asadmin create-jdbc-connection-pool --user admin --password adminadmin
--host sas.sun.com --port 8888 --instance server1 --restype
javax.sql.XADataSource --datasourceclassname
oracle.jdbc.xa.client.OracleXADataSource --description "Sample Jdbc
Connection Pool" --property
User="oracle":Password="oracle":URL="jdbc\:oracle\:thin\:@oracleserver.sun
.com\:1521\:ORA" SampleJdbcConnectionPool
```

---

<b>NOTE</b>	If you want to enable “Global Transaction Support” for the new connection pool, set <code>--restype javax.sql.XADataSource</code> . In the URL property, replace the colon (:) with (\:)
-------------	--

---

Once the JDBC connection pool is created successfully, you see the following message:.

```
Created the JDBC connection pool resource with id =
SampleJdbcConnectionPool
```

### Step 2 - Apply Changes To the Instance

Now that you have successfully created a JDBC connection pool, you need to apply the changes to the current instance of Sun Java System Application Server.

The following is the syntax for applying the changes to your instance of Sun Java System Application Server.

```
asadmin reconfig --user admin_user [--password admin_password] [--host localhost]
[--port adminport] [--secure | -s]
[--discardmanualchanges=false|--keepmanualchanges=false] instancename
```

For example, the following command applies the changes to *server1*, the instance of Sun Java System Application Server.

```
asadmin reconfig --user admin --password adminadmin --host sas.sun.com --port
8888 server1
```

Once the changes are applied to your instance of Sun Java System Application Server, you will see the following message.

```
Successfully reconfigured
```

### *Using the Command Line Interface to Manage JDBC Connection Pools*

You can use the command line interface to manage JDBC connection pools and their properties, as discussed in this section:

**Listing Connection Pools.** The following command lists all the connection pools created for *server1*, the instance of Sun Java System Application Server used in [Step 2](#).

```
asadmin list-jdbc-connection-pools --user admin --password adminadmin --host
sas.sun.com --port 8888 server1
```

**Changing a JDBC Connection Pool Property.** You can change a JDBC connection pool's property, for example, the `maxPoolSize` property, as follows:



1. Run the following command to get the value specified for the JDBC connection pool attribute `maxPoolSize`.

```
asadmin get -u admin -w adminadmin -H sas.sun.com -p 8888
server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSize
```

When you run this command, you will see the following result:

```
server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSize =
32
```

Change the value of `MaxPoolSize` to 80, by running the following command:

```
asadmin set -u admin -w adminadmin -H sas.sun.com -p 8888
server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSize="80"
```

Once the values are specified as indicated, you will see the following message:

```
Attribute maxPoolSize set to 80
```

2. Apply the changes to your instance of Sun Java System Application Server using the following command:

```
asadmin reconfig --user admin --password adminadmin --host sas.sun.com
--port 8888 server1
```

**Changing the User Property.** In the following piece of sample code, you can change the property “User” from `oracle` to `System`.

```
asadmin create-jdbc-connection-pool --user admin --password adminadmin --host
sas.sun.com --port 8888 --instance server1 --restype javax.sql.XADataSource
--datasourceclassname oracle.jdbc.xa.client.OracleXADataSource --description "Sample Jdbc
Connection Pool" --property
User="oracle":Password="oracle":URL="jdbc\:oracle\:thin\:@oracleserver.sun.com\:1521\:ORA"
SampleJdbcConnectionPool
```

1. Run the following command to change the User property.

```
asadmin set -u admin -w adminadmin -H sas.sun.com -p 8888
server1.jdbc-connection-pool.SampleJdbcConnectionPool.property.User="System"
```

The name of the user is changed from *Oracle* to *System*.

2. After you change the user name, run the following command to apply your changes:

```
asadmin reconfig --user admin --password adminadmin --host sas.sun.com
--port 8888 server1
```

**Creating a JDBC resource called `SampleJdbcResource`.** You can create a JDBC resource, as detailed below. The following is the syntax for creating a JDBC resource:

```
asadmin create-jdbc-resource --user admin_user [--password admin_password]
[--host localhost] [--port 4848] [--secure | -s] [--instance instancename]
--connectionpoolid id [--enabled=true] [--description text] [--property
(name=value)[:name=value]*] jndiname
```

1. Run the following command to create a JDBC resource called `SampleJdbcResource`.

```
asadmin create-jdbc-resource --user admin --password adminadmin --host
sas.sun.com --port 8888 --instance server1 --description "Sample Jdbc
Resource" --connectionpoolid SampleJdbcConnectionPool
jdbc/SampleJdbcResource
```

When you run this command, the JDBC resource is created, and you will see the following message:.

```
Created the external JDBC resource with jndiname =
jdbc/SampleJdbcResource
```

2. Next, you need to run the following command to apply the changes to your instance of Sun Java System Application Server.

```
asadmin reconfig --user admin --password adminadmin --host sas.sun.com
--port 8888 server1
```

3. Run the following command to list all the JDBC resources in the instance `server1`.

```
asadmin list-jdbc-resources --user admin --password adminadmin --host
sas.sun.com --port 8888 server1
```

## About Connection Pooling

Applications can obtain connections by first looking up the `DataSource` using JNDI. A sample code fragment to accomplish this, is shown below:

```
InitialContext ctx = new InitialContext();

DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/employee_ds");
```

After obtaining the `DataSource`, the application component can obtain connections in two ways depending on the value set on the `<res-auth>` element in the J2EE deployment descriptor. If the value of this element is `Container`, then the application can obtain a connection using `ds.getConnection()` method (that is, without specifying any sign-on information). Otherwise, the application has to give the sign-on information to obtain the connection from the resource manager like `ds.getConnection(userName, password)`.

All requests to `getConnection()` will be served from a pool. The JDBC Connection pool will be created based on a set of parameters described in `server.xml`. When the pool is created, the pool contains the number of connections that are available initially. So the `ds.getConnection()` request can be satisfied with the connections currently available in the pool. The next request (if none of the previous connections have been returned to the pool) will find the pool empty and will result in the creation of incremented connections, subject to the limit of the maximum connections specified for the pool. The pool implementation will keep track of the number of connections that are created. If a `getConnection()` request finds the pool empty, or if the number of created connections is equal to the maximum connections in the pool, then the current request is blocked. This happens only if connection sharing is not possible, and continues until a connection is returned to the pool.

The connection pool will continue to work properly even if the database crashes and comes up, while the server is still running. This can be accomplished only if you enable connection validation, as described in [“Configuring Connection Pool Settings” on page 251](#).

Depending on the value that you select from the Validation Type drop-down list, the following parameters are executed by the pool implementation program:

- If you choose `auto-commit` as your connection validation type, the system checks to see if the connection is valid by performing a `conn.getAutoCommit()` method. If the method does not throw an `SQLException` then the `Connection` is assumed to be valid. `auto-commit` is the default option for this parameter.
- If you chose `meta-data` as your connection validation type, then the `conn.getMetaData()` method will be performed, to check for metadata on the connection. and if the method didn't throw `SQLException` then the `Connection` is assumed to be valid.
- If you choose `table` as your connection validation type, then the query “`Select * From <table-name>`” will be executed. If this call doesn't throw an `SQLException` then the connection is assumed to be valid.

If you enable the fail-all-connections property, then if any connection in the pool is found to be invalid, all connections will be closed and re-established. Otherwise, invalidation and re-establishment is lazy and happens on use of individual connections.

The pool implementation also provides the ability to recycle all available connections in the pool. So if connections are idle beyond the idle period specified, they will be closed, bringing the pool size to the steady pool size. If the pool is extremely idle, it may result in the container having to re-establish stale connection and always keep the pool populated with a steady pool of usable connections. You must keep this in mind while determining whether to set the steady pool size relative to the maximum pool size.

### Monitoring JDBC Connection Pooling

You may want to monitor the pool operation periodically, to determine if the pool size configuration is working effectively. Using the command line interface, the following aspects of JDBC connection pool operation can be watched. The following table lists all the JDBC connection pooling parameters that can be monitored.

Note that the mechanics of enabling monitoring, and the attributes that can be monitored are likely to be improved in forthcoming releases.

**Table 9-6**    JDBC Connection Pool Parameters for Monitoring

Attribute Name	Data Type	Description
total-threads-waiting	Integer	Total threads waiting for JDBC connection.
total-outbound-connections	Integer	Total JDBC connection validation failures
total-connections-timed-out	Integer	Total connection requests timed out.

### About Connection Sharing

When multiple connections acquired by a J2EE application use the same resource manager, the pool implementation will provide connection sharing within the same transaction scope. The term transaction scope, can be understood from the following example:

Bean\_A starts a transaction (Tx1) and obtains a connection. Bean\_A then calls a method in Bean\_B in the same transaction (Tx1). Now if Bean\_B acquires a connection from the same `DataSource` and with the same sign-on information, it is clear that the same connection can be shared as only Bean\_A will complete the transaction. Also, note that connections will be shared only if the resource sharing scope is set to `Shareable` in the J2ee deployment descriptor. If connection sharing is not desired, then the resource sharing scope must be set to `Unshareable`, in the deployment descriptor. Sun Java System Application Server provides connection sharing, as it provides better performance.

## About JDBC Transactions

A transaction consists of one or more statements that have been executed, completed, and then either committed or rolled back. When the method `commit` or `rollback` is called, the current transaction ends and another one begins.

Generally a new `Connection` object is in auto-commit mode by default, meaning that when a statement is completed, the method `commit` will be called on that statement automatically. In this case, since each statement is committed individually, a transaction consists of only one statement. If the auto-commit mode has been disabled, a transaction will not terminate until the method `commit` or `rollback` is called explicitly, so it will include all the statements that have been executed since the last invocation of either `commit` or `rollback`. In this second case, all the statements in the transaction are committed or rolled back as a group.

The method `commit` makes permanent any changes an SQL statement makes to a database, and it also releases any locks held by the transaction. The method `rollback` will discard those changes.

In the case of two updates in one transaction, you might at times not want one change to take effect in an update, unless the other update is also affected. This can be accomplished by disabling auto-commit and grouping both updates into one transaction. If both updates are successful, then the `commit` method is called, making the effects of both updates permanent; if one fails or both fail, then the `rollback` method is called, restoring the values that existed before the updates were executed. Most JDBC drivers support transactions.

Classes and interfaces in the `javax.sql` package make it possible for `Connection` objects to be part of a distributed transaction, that is, a transaction that involves connections to more than one DBMS server. In order to be able to participate in distributed transactions, a `Connection` object must be produced by a `DataSource` object that has been implemented to work with the middle tier server's distributed transaction infrastructure. Unlike `Connection` objects produced by the

`DriverManager`, a `Connection` object produced by such a `DataSource` object will have its auto-commit mode disabled by default. A standard implementation of a `DataSource` object, on the other hand, will produce `Connection` objects that are exactly the same as those produced by the `DriverManager` class.

When a `Connection` object is part of a distributed transaction, a transaction manager determines when the methods `commit` or `rollback` are called on it. Thus, when a `Connection` object is participating in a distributed transaction, an application should not do anything that affects when a connection begins or ends, such as calling the methods `Connection.commit` or `Connection.rollback`, or turning on the connection's auto-commit mode. These would interfere with the transaction manager's handling of the distributed transaction.

## About Java Mail Resources

The JavaMail API allows for access to email messages contained in message stores, and for the creation and sending of email messages using a message transport. Specific support is included for Internet standard MIME messages. Access to message stores and transports is through protocol providers supporting specific store and transport protocols. The JavaMail API specification does not require any specific protocol providers, but JavaMail includes an IMAP message store provider and an SMTP message transport provider.

The JavaMail API provides a set of abstract classes defining objects that comprise a mail system. The API defines classes like `Message`, `Store` and `Transport`. The API can be extended and can be sub-classed to provide new protocols and to add functionality when necessary. In addition, the API provides concrete sub-classes of the abstract classes. These sub-classes, including `MimeMessage` and `MimeBodyPart`, implement widely used Internet mail protocols.

The JavaMail API draws heavily from IMAP, MAPI, CMC, c-client and other e-mail messaging system APIs. The JavaMail API supports many different messaging system implementation like, different message stores, different message formats, and different message transports. The JavaMail API provides a set of base classes and interfaces that define the API for client applications. Developers can sub-class JavaMail classes to provide the implementations of particular messaging systems, such as IMAP, POP3, and SMTP.

The following topics are covered in this section:

- [About the JavaMail Message-handling Process](#)
- [About the Architectural Components of JavaMail](#)

- [About JavaBeans Activation Framework \(JAF\)](#)
- [About JavaMail Configuration Parameters](#)
- [J2EE Deployment Descriptor for JavaMail Session References](#)
- [Entries in Application Server Deployment Descriptor](#)
- [Creating a New JavaMail Session](#)
- [Configuring Advanced Resource Properties](#)

## About the JavaMail Message-handling Process

The JavaMail API performs the following functions, which comprise the standard mail handling process for a typical client application:

- Create a mail message consisting of a collection of header attributes and a block of data of some known data type as specified in the Content-Type header field. JavaMail uses the Part interface and the Message class to define a mail message. It uses the JAF-defined DataHandler object to contain data placed in the message.
- Create a Session object, which authenticates the user, and controls access to the message store and transport.
- Send the message to its recipient list.
- Retrieve a message from a message store.
- Execute a high-level command on a retrieved message. High-level commands like view and print are intended to be implemented via JAF-Aware JavaBeans.

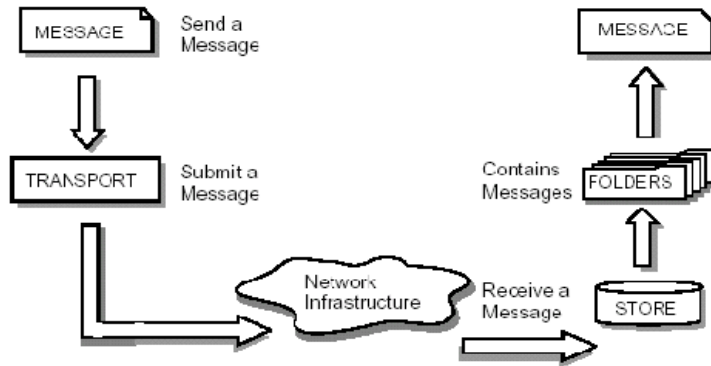
---

<b>NOTE</b>	Currently, the JavaMail framework does not define mechanisms that support message delivery, security, disconnected operation, directory services or filter functionality.
-------------	---

---

The following figure illustrates how the JavaMail API handles the message handling process:

**Figure 9-3** Message Handling Process of the Java Mail API



The JavaMail API is configured by using a static factory method to create a `javax.mail.Session`. Sun Java System Application Server requests a Session object using JNDI, and lists its need for a Session object in its deployment descriptor using a `resource-ref` element. A JavaMail API Session object is considered a resource factory.

Message transport is provided that is capable of handling addresses of type `javax.mail.internet.InternetAddress` and messages of type `javax.mail.internet.MimeMessage`. The default message transport must be properly configured to send such messages using the `send` method of the `javax.mail.Transport` class.

The Abstract Layer of the JavaMail API declares classes, interfaces and abstract methods intended to support mail handling functions that all mail systems support. API elements comprising the Abstract Layer are intended to be sub-classed and extended as necessary in order to support standard data types, and to interface with message access and message transport protocols as necessary.

The internet implementation layer implements part of the abstract layer using internet standards - RFC822 and MIME.



# About the Architectural Components of JavaMail

This section describes the major components comprising the JavaMail architecture, given in the following topics:

- [Message Class](#)
- [Message Storage and Retrieval](#)
- [Message Composition and Transport](#)

## Message Class

The Message class is an abstract class that defines a set of attributes and a content for a mail message. Attributes of the Message class specify addressing information and define the structure of the content, including the content type. The content is represented as a DataHandler object that wraps around the actual data.

The Message class implements the Part interface. The Part interface defines attributes that are required to define and format data content carried by a Message object, and to interface successfully to a mail system. The Message class adds From, To, Subject, Reply-To, and other attributes necessary for message routing via a message transport system. When contained in a folder, a Message object has a set of flags associated with it. JavaMail provides Message subclasses that support specific messaging implementations.

The content of a message is a collection of bytes, or a reference to a collection of bytes, encapsulated within a Message object. JavaMail has no knowledge of the data type or format of the message content. A Message object interacts with its content through an intermediate layer—the JavaBeans Activation Framework (JAF). This separation allows a Message object to handle any arbitrary content and to transmit it using any appropriate transmission protocol by using calls to the same API methods. The message recipient usually knows the content data type and format and knows how to handle that content.

The JavaMail API also supports multipart Message objects, where each Bodypart defines its own set of attributes and content.

## Message Storage and Retrieval

Messages are stored in Folder objects. A Folder object can contain sub-folders as well as messages, thus providing a tree-like folder hierarchy. The Folder class declares methods that fetch, append, copy and delete messages. A Folder object can also send events to components registered as event listeners.

### *The Store Class*

The Store class defines a database that holds a folder hierarchy together with its messages. The Store class also specifies the access protocol that accesses folders and retrieves messages stored in folders. The Store class also provides methods to establish a connection to the database, to fetch folders and to close a connection. Service providers implementing Message Access protocols (IMAP, POP3 etc.) start off by sub-classing the Store class. A user typically starts a session with the mail system by connecting to a particular Store implementation.

### Message Composition and Transport

A client creates a new message by instantiating an appropriate Message subclass. It sets attributes like the recipient addresses and the subject, and inserts the content into the Message object. Finally, it sends the Message by invoking the Transport.send method. The Transport class models the transport agent that routes a message to its destination addresses. This class provides methods that send a message to a list of recipients. Invoking the Transport.send method with a Message object identifies the appropriate transport based on its destination addresses.

### *The Session Class*

The Session class defines global and per-user mail-related properties that define the interface between a mail-enabled client and the network.

JavaMail system components use the Session object to set and get specific properties. The Session class also provides a default authenticated session object that desktop applications can share. The Session class is a final concrete class. It cannot be sub-classed. The Session class also acts as a factory for Store and Transport objects that implement specific access and transport protocols. By calling the appropriate factory method on a Session object, the client can obtain Store and Transport objects that support specific protocols.

## About JavaBeans Activation Framework (JAF)

JavaMail uses the JavaBeans Activation Framework (JAF) in order to encapsulate message data, and to handle commands intended to interact with that data. Interaction with message data should take place via JAF-aware JavaBeans, which are not provided by the JavaMail API.

With the JavaBeans Activation Framework standard extension, developers who use Java technology can take advantage of standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and to instantiate the appropriate bean to perform said operation(s).

For example, if a browser obtained a JPEG image, this framework would enable the browser to identify that stream of data as an JPEG image, and from that type, the browser could locate and instantiate an object that could manipulate, or view that image.

The JavaBeans Activation Framework API supports various MIME data types. The JavaMail API must include `javax.activation.DataContentHandlers` for the following MIME data types corresponding to the Java programming language type indicated in the following table.

**Table 9-7** JavaMail API MIME Data Type to Java Type Mappings

MIME Type	Java Type
Text/Plain	<code>java.lang.String</code>
Multipart/	<code>javax.mail.internet.MIME.Multipart</code>
Message/rfc822	<code>javax.mail.internet.MIME.Message</code>

The JavaBeans Activation Framework integrates support for MIME data types into the Java platform. MIME byte streams can be converted to and from Java programming language objects, using `avax.activation.DataContentHandler` objects. JavaBeans components can be specified for operating on MIME data, such as viewing or editing the data. The JavaBeans Activation Framework also provides a mechanism to map filename extensions to MIME types. The JavaBeans Activation Framework is used by the JavaMail API to handle the data included in email message. Typical J2EE applications will not need to use the JavaBeans Activation Framework directly, although applications making sophisticated use of email may need it.

## About JavaMail Configuration Parameters

The following configuration parameters are used by JavaMail Resources in Sun Java System Application Server. These configuration parameters are name, value pairs that would be read from the `mail-resource` element of the `server.xml` file.

- JNDI Name

The JNDI Name specifies the name with which this mail resource will be referenced from a J2EE Application.

- **Enabled**

The `enabled` configuration parameter specifies whether this mail resource will be published in the JNDI tree and can be referenced. If a J2EE application references a resource which is disabled, it will receive a `NameNotFoundException` exception.

- **store-protocol**

Specifies the default Message Access Protocol. The `Session.getStore()` method returns a `Store` object that implements this protocol. The client can override this property and explicitly specify the protocol with the `Session.getStore(String protocol)` method.

- **store-protocol class**

Specifies the class name that implements the store protocol specified above. The default for this class is `com.sun.mail.imap.IMAPStore`.

- **transport-protocol**

Specifies the default Transport Protocol. The `Session.getTransport()` method returns a `Transport` object that implements this protocol. The client can override this property and explicitly specify the protocol by using `Session.getTransport(String protocol)` method.

- **transport-protocol class**

Specifies the class name that implements the transport protocol specified above. The default for this class is `com.sun.mail.smtp.SMTPTransport`.

- **host**

Specifies the default Mail server. The `Store` and `Transport` object's connect methods use this property, if the protocol specific host property is absent, to locate the target host.

- **user**

Specifies the username to provide when connecting to a Mail server. The `Store` and `Transport` object's connect methods use this property, if the protocol specific username property is absent, to obtain the username.

- **from**

Specifies the return address of the current user. Used by the `InternetAddress.getLocalAddress` method to specify the current user's email address.

- `debug`

Specifies the initial debug mode. Setting this property to true will turn on debug mode, while setting it to false turns debug mode off.

- `mail-<protocol>-host`

Specifies the protocol-specific default Mail server. This overrides the `mail.host` property. This property could be set depending on the value of the `store-protocol` attribute. For the values of `imap` or `POP` for `store-protocol`, you need to add properties with the name `mail.imap.host` or `mail.pop3.host`, respectively. The value for the specific property should be set as it is appropriate in your mail system configuration. For example, with `store-protocol` set to `IMAP`, the property name: `mail-imap-host` would bear the value: `spaceduck.acme.com`.

- `mail-<protocol>-user`

Specifies the protocol-specific default user name for connecting to the Mail server. This overrides the `mail.user` property. So this property could be `mail.imap.user` or `mail.pop3.user` depending on the selection of `store-protocol` attribute. For example, with `store-protocol` set to `IMAP`, the property name: `mail-imap-user` would bear the value `fredbloggs`.

## J2EE Deployment Descriptor for JavaMail Session References

Once a JavaMail Resource is registered with the server, it can be referenced in any J2EE application component using a JNDI lookup. In order to deploy an application that references resource manager connection factories, the component provider must declare all the resource manager connection factory references in the standard J2EE 1.3 deployment descriptor.

The complete J2EE1.3 descriptor elements for JavaMail reference are described below:

```
<resource-ref>
  <description>
    JavaMail resource used for sending my mail
  </description>
  <res-ref-name>mail/MyMailSession</res-ref-name>
  <res-type>javax.mail.Session</res-type>
```

```

    <res-auth>Container</res-auth>

    <res-sharing-scope>Shareable</res-sharing-scope>

</resource-ref>

```

## Entries in Application Server Deployment Descriptor

For each deployed component that refers to a mail resource, the deployer must map the resource name used in the component to the actual jndi name with which the DataSource is registered with the naming service. The deploy tool is expected to help the deployer to easily make this mapping. This mapping is registered in the Sun Java System Application Server specific xml. The fragment of the Sun Java System Application Server specific XML containing the mapping is shown below:

```

<resource-ref>

    <res-ref-name>mail/MyMailSession</res-ref-name>

    <jndi-name>mail/Session</jndi-name>

</resource-ref>

```

## Creating a New JavaMail Session

You can configure JavaMail sessions using the Administration interface. To create and configure a new JavaMail session, perform the following tasks:

1. In left pane of the Administration interface, expand the Sun Java System Application Server instance for which you want to create a new JavaMail session.
2. Click JavaMail Sessions.
3. Click New. You see the page for creating a JavaMail resource in the right pane.
4. In the JNDI Name text field, provide a JNDI name for the Java Mail session that you are creating. Once a Java Mail Resource is registered with the server, it can be referenced in any J2EE application component using a JNDI lookup.
5. In the Mail Host text field, specify the DNS name of the default mail server. The connect methods of the Store and Transport objects use this property, to locate the target host, if the protocol-specific host property is absent.

6. In the Default User text field, specify the username to provide when connecting to a Mail server. The connect methods of the Store and Transport objects use this property to obtain the username, if the protocol-specific username property is absent.
7. In the Default Return Address field, specify the default return address of the current user. The default address should be in this form: username@host.
8. In the Description field, provide a description for this Java Mail session.
9. Mark the Java Mail Session Enabled checkbox to enable the Java Mail session that you created.
10. Click OK to save the new JavaMail session that you have configured.

## Configuring Advanced Resource Properties

You can configure several additional properties for your new JavaMail session using the Administration interface. The property name and value pairs depend upon the mail protocol in use. You can also directly specify these properties in the `server.xml` file.

To configure additional properties, perform the following tasks:

1. In left pane of the Administration interface, expand the Sun Java System Application Server instance for which you want to modify the JavaMail session configuration.
2. Click JavaMail Sessions.
3. Click the name of the JavaMail Session you want to edit.
4. In the Store Protocol text field, specify the store protocol that you would like to use for this specific JavaMail session, for example, POP3, or IMAP.
5. In the Store Protocol Class text field, specify the class name of the store protocol that you have indicated, as shown in the given example.
6. In the Transport Protocol text field, specify the transport protocol that you want to use for this specific session of JavaMail, for example, SMTP.
7. In the Transport Protocol Class text field, specify the class name of the transport protocol you have indicated for this session, as shown in the above example.
8. Mark the Debug Enabled checkbox, to enable debugging of this specific session of JavaMail. Enabling this checkbox turns the debug mode on.

**9. Click OK to save your additional properties configuration.**

A complete sample of mail resource configuration is shown below:

```
<mail-resource
  jndi-name = "mail/Session"
  enabled = "true"
  store-protocol = "imap"
  store-protocol-class = "com.sun.mail.imap.IMAPStore"
  transport-protocol = "smtp"
  transport-protocol-class = "com.sun.mail.smtp.SMTPTransport"
  host = "gopostal.acme.com"
  user = "kingkong"
  from = "kingkong@acme.com"
  debug = "false">
  <property name = "mail-imap-host" value = "spaceduck.acme.com"/>
  <property name = "mail-imap-user" value = "fredbloggs"/>
</mail-resource>
```



# Using the JMS Service

Sun Java System Application Server provides support for applications that use the Java Message Service (JMS) application programming interface (API) for messaging operations. JMS is a set of programming interfaces that provide a common way for Java applications to create, send, receive, and read messages in a distributed environment.

In particular, JMS is the standards-based way that Java 2 Enterprise Edition (J2EE) applications perform asynchronous messaging. Accordingly, J2EE components—Web components or Enterprise JavaBeans (EJB) components—can use the JMS API to send messages that can be consumed asynchronously by a specialized EJB, called a message-driven bean (MDB).

Sun Java System Application Server support for JMS messaging, in general, and for MDBs, in particular, requires messaging middleware that implements the JMS specification—a JMS provider. Sun Java System Application Server uses Sun Java System Message Queue 3.5 Service Pack 1 as its native JMS provider.

MQ is tightly integrated into Sun Java System Application Server, providing transparent JMS messaging support. This support (known within Sun Java System Application Server as the *JMS Service*) requires only minimal administration.

This chapter provides information needed to understand and administer the built-in JMS Service provided through Sun Java System Message Queue. It includes the following sections:

- [About JMS](#)
- [Built-in JMS Service](#)
- [Administration of the Built-in JMS Service](#)

# About JMS

The JMS specification describes a set of programming interfaces that support distributed, enterprise messaging. An enterprise messaging system enables independent distributed components or applications to interact through messages. These components, whether on the same system, the same network, or loosely connected through the Internet, use messaging to pass data and coordinate their respective functions.

To support enterprise-scale messaging, JMS provides for reliable, asynchronous message delivery.

**Reliable delivery.** Messages from one component to another must not be lost due to network or system failure. This means the system must be able to guarantee that a message is successfully delivered.

**Asynchronous delivery.** For large numbers of components to be able to exchange messages simultaneously, and support high density throughputs, the sending of a message cannot depend upon the readiness of the consumer to immediately receive it. If a consumer is busy or offline, the system must allow for a message to be sent and subsequently received when the consumer is ready. This is known as asynchronous message delivery, popularly known as store-and-forward messaging.

This topic is a brief overview of JMS concepts and terminology:

- [Basic Messaging System Concepts](#)
- [JMS Specification](#)
- [Message-driven Beans](#)

For a full description of JMS, see the JMS 1.0.2 specification, which can be found at the following location:

<http://java.sun.com/products/jms/docs.html>

## Basic Messaging System Concepts

A few basic concepts underlie enterprise messaging systems in general, and JMS in particular. These are discussed in this topic.

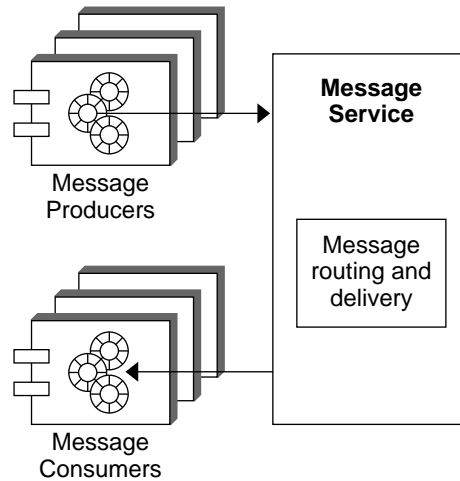
## Message

A message consists of data in some format (message body) and meta-data that describes the characteristics or properties of the message (message header), such as its destination, lifetime, or other characteristics determined by the messaging system.

## Message Service Architecture

The basic architecture of a messaging system is illustrated in [Figure 10-1](#). It consists of message producers and message consumers that exchange messages by way of a common message service. In general, any number of message producers and consumers can reside in the same messaging component. A message producer sends a message to a message service. The message service, in turn, using message routing and delivery components, delivers the message to one or more message consumers that have registered an interest in the message. The message routing and delivery components are responsible for guaranteeing delivery of the message to all appropriate consumers.

**Figure 10-1** Message Service Architecture



## Message Delivery Models

There are many relationships between producers and consumers: one to one, one to many, and many to many relationships. For example, you might have messages delivered from:

- one producer to one consumer
- one producer to many consumers
- many producers to one consumer
- many producers to many consumers.

These relationships are often reduced to two message delivery models: *point-to-point* and *publish/subscribe* messaging. The focus of the point-to-point delivery model is on messages that originate from a specific producer and are received by a specific consumer. The focus of publish/subscribe delivery model is on messages that originate from any of a number of producers and are received by any number of consumers. These message delivery models can overlap.

Historically, messaging systems supported various combinations of these two delivery models. The JMS API was intended to create a common programming approach that supports both point to point and publish/subscribe delivery models.

## JMS Specification

JMS specifies a message structure, a programming model, and a set of rules and semantics that govern messaging operations.

### JMS Message Structure

According to the JMS specification, a message is composed of three parts: a header, properties (which can be thought of as an extension of the header), and a body.

**Header.** The header specifies the JMS characteristics of the message: its destination, whether is persistent or not, its time to live, and its priority. These characteristics govern how the messaging system delivers the message.

**Properties.** Properties are optional—they provide values that applications can use to filter messages according to various selection criteria.

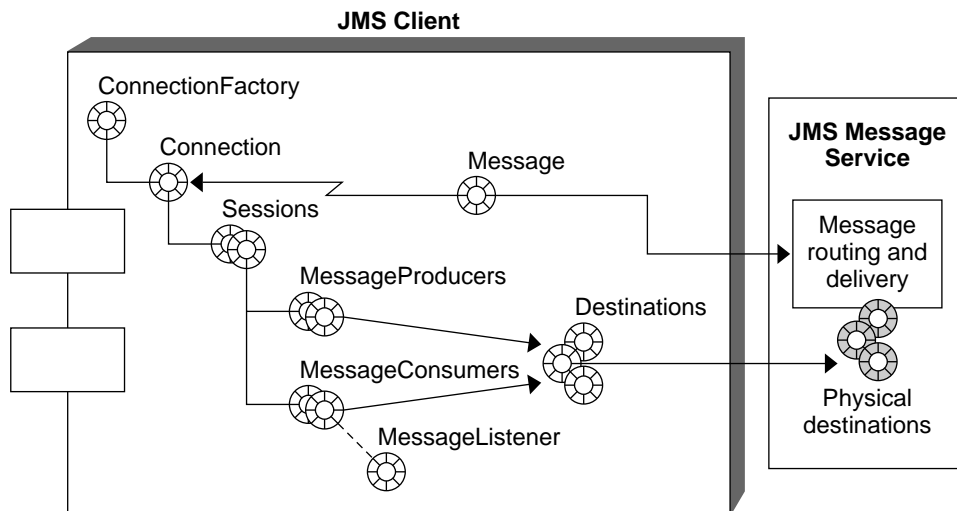
**Message body.** The message body contains the actual data to be exchanged. JMS supports six body types.

## JMS Programming Model

In the JMS programming model, JMS clients (components or applications) exchange messages by way of a JMS message service. Message producers send messages to the message service, from which message consumers receive them. These messaging operations are performed using a set of objects (furnished by a JMS provider) that implement the JMS API. [Figure 10-2](#) shows the JMS objects used to program the delivery of messages.

In the JMS programming model, a JMS client uses a `ConnectionFactory` object to create a connection over which messages are sent to and received from the JMS message service. A `Connection` is a JMS client's active connection to the message service. Both allocation of communication resources and authentication of the client take place when a connection is created.

**Figure 10-2** JMS Programming Objects



The connection is used to create sessions. A `Session` is a single-threaded context for producing and consuming messages. It is used to create the message producers and consumers that send and receive messages. A session supports reliable delivery through a number of acknowledgement options or through transactions (which can be managed by a distributed transaction manager).

A JMS client uses a `MessageProducer` to send messages to a specified physical destination, represented in the API as a destination object. The message producer can specify a default delivery mode (persistent vs. non-persistent messages), priority, and time-to-live that govern all messages sent by the producer to the physical destination.

Similarly, a JMS client uses a `MessageConsumer` to receive messages from a specified physical destination, represented in the API as a destination object. A message consumer can support either synchronous or asynchronous consumption of messages. Asynchronous consumption is achieved by registering a `MessageListener` with the consumer. The client consumes a message when a session thread invokes the `onMessage()` method of the `MessageListener` object.

## Administered Objects: Provider Independence

Two of the objects described in [Figure 10-2 on page 277](#) depend on the details of how a JMS provider implements a JMS message service. The connection factory object depends on the underlying protocols and mechanisms used by the provider to deliver messages, and the destination object depends on the specific naming conventions and capabilities of the physical destinations used by the provider.

Normally these provider-specific characteristics would make JMS client code dependent on the details of a JMS API implementation. To make JMS client code provider-independent, however, the JMS specification requires that provider-specific objects—called administered objects—be accessed in a standardized way, rather than directly instantiated in client code.

Administered objects encapsulate provider-specific implementation and configuration information. They are created and configured by an administrator, stored in a name service, and accessed by client applications through standard JNDI lookup code. Using administered objects in this way makes JMS client code provider-independent.

JMS provides for two general types of administered objects: connection factories and destinations. Both encapsulate provider-specific information, but they have very different uses within a JMS client. A connection factory is used to create connections to a message server, while destination objects are used to identify physical destinations used by the JMS message service.

---

<b>NOTE</b>	In the context of the Sun Java System Application Server, JMS administered objects are regarded as JMS resources, similar to other Sun Java System Application Server resources.
-------------	--

---

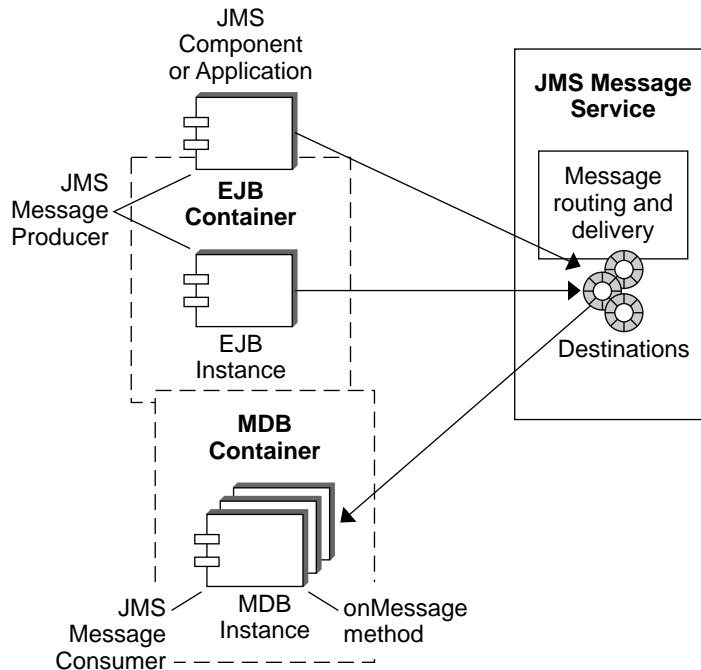
## Message-driven Beans

In addition to the general JMS client programming model introduced in “[JMS Programming Model](#)” on page 277, there is a more specialized adaptation of the JMS API used in the context of J2EE applications. This specialized JMS client is called a *message-driven bean* and is one of a family of EJB components specified in the EJB 2.0 Specification (<http://java.sun.com/products/ejb/docs.html>).

The need for message-driven beans arises out of the fact that other EJB components (session beans and entity beans) can only be called synchronously. Hence, when you invoke a method of these beans, resources are blocked until the methods have completed. These EJB components have no mechanism for receiving messages asynchronously, since they are only accessed through standard EJB interfaces.

However, asynchronous messaging is a requirement of many enterprise applications. Hence, the need for an EJB component that can receive messages and consume them without being tightly coupled to the producer of the message.

The MDB is a specialized EJB component supported by a specialized EJB container (a software environment that provides distributed services for the components it supports).

**Figure 10-3** MDB Message Consumers

**Message-driven Bean.** The MDB is a JMS message consumer that implement the `JMS MessageListener` interface. Its `onMessage` method (written by the MDB developer) is invoked when a message is received by the MDB container. The `onMessage` method consumes the message, just as the `onMessage` method of any JMS `MessageListener` object would. The MDB can consume messages from a single destination. The messages can be produced by standalone JMS client applications, Web components, or other EJB components, as shown in [Figure 10-3](#).

**MDB Container.** The MDB is supported by a specialized EJB container, responsible for creating instances of the MDB and setting them up for asynchronous consumption of messages. This involves setting up a connection with the message service (including authentication), creating a pool of sessions associated with a given destination, and managing the distribution of messages as they are received among the pool of sessions and associated MDB instances. Since the container controls the life-cycle of MDB instances, it manages the pool of MDB instances so as to accommodate incoming message loads.



Associated with an MDB is a deployment descriptor that specifies the JNDI lookup names for the administered objects used by the container in setting up message consumption: a connection factory and a destination. The deployment descriptor might also include other information that can be used by deployment tools to configure the container. Each such container supports instances of only a single MDB.

For information on configuring an EJB container's MDB settings for the Sun Java System Application Server, see [“About Message-driven Beans” on page 191](#).

## Built-in JMS Service

Support for JMS messaging, in general, and for MDBs, in particular, is built into Sun Java System Application Server. This support is achieved through the tight integration of Sun Java System Message Queue with the Sun Java System Application Server, providing a native, built-in JMS Service.

This topic covers the following topics necessary to understand this built-in JMS Service:

- [About Sun Java System Message Queue](#)
- [Integration of MQ with Sun Java System Application Server](#)

For information on administering the built-in JMS Service, see [“Administration of the Built-in JMS Service” on page 290](#).

## About Sun Java System Message Queue

Sun Java System Message Queue (MQ) is an enterprise messaging system that implements the JMS open standard: it is a JMS provider.

The MQ product has features that go beyond the minimum requirements of the JMS specification for reliable, asynchronous messaging. Some of these features such as the following are available in the MQ Platform Edition integrated into Sun Java System Application Server:

- centralized administration
- tunable performance
- support for multiple messaging transports
- user authentication and authorization

The following additional features are available by upgrading to the MQ Enterprise Edition:

- support for multi-broker message services
- HTTP/HTTPS connections
- secure connection services
- scalable connection capability
- multiple queue delivery policies

You can, however, run the MQ broker with a trial license of the Enterprise Edition of MQ. The trial license has a limited 90-day duration enforced by the software, making it suitable for evaluating the enterprise features available in the Enterprise Edition of MQ.

For information on how to run the trial version of MQ Enterprise Edition see [Appendix D, “Running Message Queue Broker with Trial Enterprise License.”](#) The Appendix also provides information on how to purchase the Enterprise Edition, should you wish to do so.

## Components of an MQ Messaging System

An MQ messaging system consists of a number of parts, as illustrated in [Figure 10-4 on page 283](#), that work together to provide for reliable message delivery.

The main parts of an MQ messaging system are the following:

- [MQ Message Server](#)
- [MQ Client Runtime](#)
- [MQ Administered Objects](#)
- [MQ Administration Tools](#)

These are introduced briefly in the following topics. For a more complete discussion of the MQ messaging system, see the *Sun Java System Message Queue Administration Guide*, which can be found at the following location:

<http://docs.sun.com/>

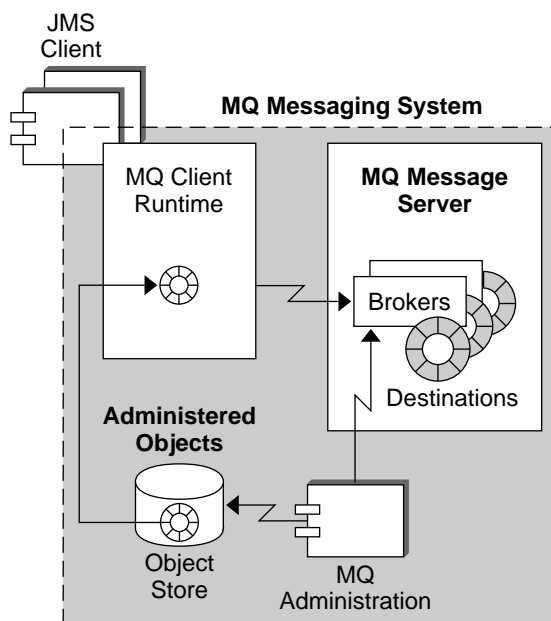
## MQ Message Server

The main parts of an MQ message server, as shown in [Figure 10-4](#), are brokers and physical destinations.

**Brokers.** A broker provides delivery services for an MQ messaging system. Message delivery relies upon a number of supporting components that handle connection services, message routing and delivery, persistence, security, and logging. A message server can employ one or more brokers to achieve scalability.

**Physical Destinations.** Delivery of a message is a two-phase process—delivery from a producing client to a physical destination maintained by a broker, followed by delivery from the destination to one or more consuming clients. Physical destinations represent locations in a broker's physical memory and/or persistent storage (see [“Physical Destinations”](#) on [page 284](#) for more information).

**Figure 10-4** MQ System Architecture



## *Brokers*

Message delivery in an MQ messaging system—from producing clients to destinations, and then from destinations to one or more consuming clients—is performed by a broker (or, in the Enterprise Edition, a cluster of brokers working in tandem). To perform message delivery, a broker must set up communication channels with clients, perform authentication and authorization, route messages appropriately, guarantee reliable delivery, and provide data for monitoring system performance.

To perform this complex set of functions, a broker uses a number of different components, each with a specific role in the delivery process. You can configure these internal components to optimize the performance of the broker, depending on load conditions, application complexity, and so on. See the *MQ Administration Guide* for more information.

## *Physical Destinations*

MQ messaging is premised on a two-phase delivery of messages: first, delivery of a message from a producer client to a destination on the broker, and second, delivery of the message from the destination on the broker to one or more consumer clients. There are two types of destinations: queues (point to point delivery model) and topics (publish/subscribe delivery model). These destinations represent locations in a broker's physical memory where incoming messages are marshaled before being routed to consumer clients.

You generally create physical destinations using administration tools, however destinations can also be automatically created by the broker upon receipt of a message.

**Queue Destinations.** Queue destinations are used in point to point messaging, where a message is meant for ultimate delivery to only one of a number of consumers that has registered an interest in the destination. As messages arrive from producer clients, they are queued and delivered to a consumer client.

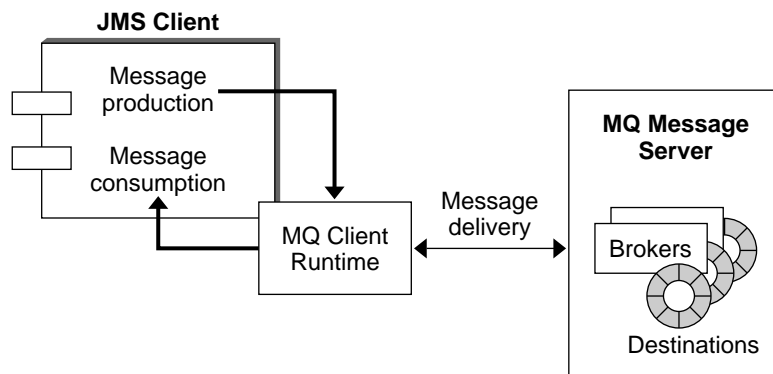
**Topic Destinations.** Topic destinations are used in publish/subscribe messaging, where a message is meant for ultimate delivery to all of the consumers that have registered an interest in the destination. As messages arrive from producers, they are routed to all consumers subscribed to the topic. If consumers have registered a durable subscription to the topic, they do not have to be active at the time the message is delivered to the topic—the broker will store the message until the consumer is once again active, and then deliver the message.

## MQ Client Runtime

The MQ client runtime provides JMS clients (stand-alone applications, Web components or EJB components) with an interface to the MQ message server—it supplies JMS clients with implementations of all the programming interfaces needed for clients to send messages to destinations and to receive messages from such destinations.

Figure 10-5 illustrates how message production and consumption involve an interaction between JMS clients and the MQ client runtime, while message delivery involves an interaction between the MQ client runtime and the MQ message server.

**Figure 10-5** Messaging Operations



## MQ Administered Objects

MQ administered objects allow JMS client code to be provider-independent (see [“Administered Objects: Provider Independence” on page 278](#)). They do this by encapsulating provider-specific implementation and configuration information in objects that can then be used by client applications in a provider-independent way. MQ administered objects are created and configured by an administrator, stored in a name service, and accessed by JMS clients through standard JNDI lookup code.

**Connection Factory Administered Objects.** A connection factory object is used to create physical connections between a JMS client (stand-alone applications, Web components or EJB components) and an MQ message server. A connection factory object has no physical representation in a broker—it is used simply to enable a JMS

client to establish connections with a broker. It is also used to specify behaviors of the connection and of the client runtime that uses the connection to access a broker: an MQ connection factory therefore has a significant number of configurable attributes that allow you to tune the performance of an MQ system.

**Destination Administered Objects.** A destination administered object (a queue or a topic) represents a physical destination (a physical queue or a physical topic) in a broker to which the publicly-named destination administered object corresponds. By creating a destination administered object, you allow JMS clients (message consumers and/or message producers) to access the corresponding physical destination.

## MQ Administration Tools

MQ administration tools fall into two categories: command line utilities and a graphical user interface (GUI) Administration Console.

**The Administration Console.** You can use the administration console to connect to a broker and manage it, create physical destinations on the broker, connect to an object store and add, update, or delete administered objects from the object store. There are some tasks that you cannot use the Administration Console to perform; chief among these are starting up a broker, creating broker clusters, configuring more specialized properties of a broker, and managing a user database.

**Command Line Utilities.** You use the MQ utilities to perform all the tasks you can perform using the administration console, and in addition, to start up and manage a broker, configure more specialized properties of a broker, and manage a MQ user database.

## Integration of MQ with Sun Java System Application Server

MQ Platform Edition is automatically installed as part of the Sun Java System Application Server installation process. For more information, see the *Sun Java System Application Server Installation Guide*.

This installation provides Sun Java System Application Server with a JMS messaging system that supports any number of application server instances. Each server instance, by default, has an associated built-in JMS Service that supports all JMS clients running in the instance.

This topic covers the following topics:

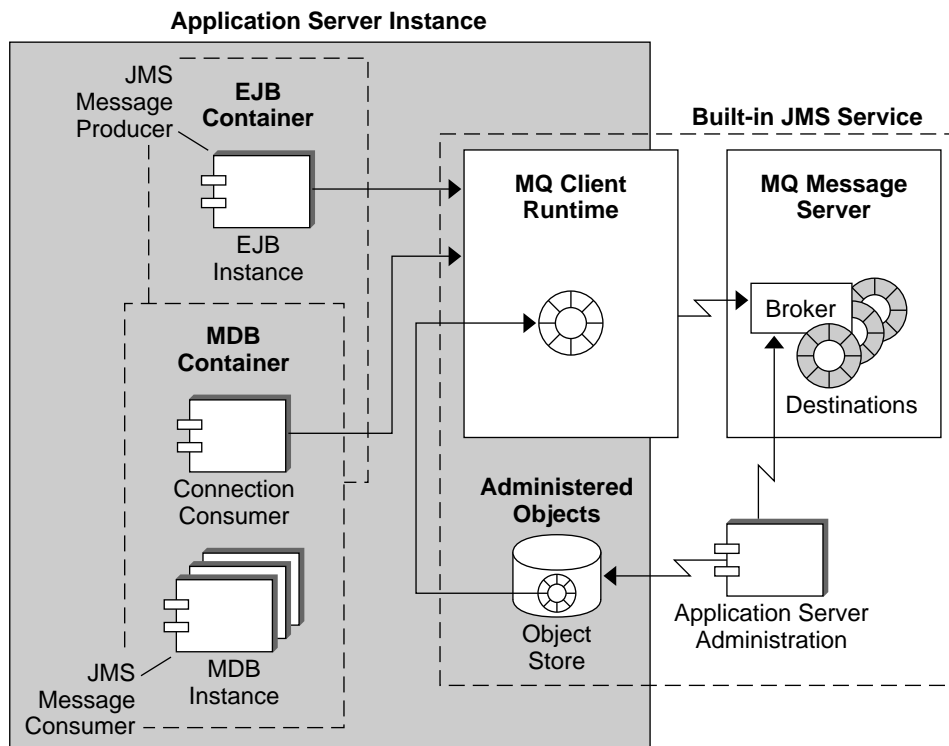
- [Architecture of the Built-in JMS Service](#)
- [Disabling the Built-in JMS Service](#)

The built-in JMS Service can be administered using Sun Java System Application Server administration tools (see [“Administration of the Built-in JMS Service”](#) on page 290).

## Architecture of the Built-in JMS Service

The built-in JMS Service—illustrated in [Figure 10-6](#)—is similar to an ordinary MQ messaging system (shown in the figure [“MQ System Architecture”](#) on page 283), except for a number of qualifications, described below.

**Figure 10-6** Built-in MQ Messaging System



**MQ Message Server.** Each application server instance is associated with its own built-in JMS Service. The built-in JMS Service makes use of a single-broker message server. The broker runs in a separate process outside of the application server instance as shown in [Figure 10-6 on page 287](#). By default, the broker instance (the built-in JMS Service) starts up when its associated server instance is started up and shuts down when the server instance is shut down. Configuration information for a server instance's built-in JMS Service is recorded in the application server configuration store (the `server.xml` file) and can be modified as described in [“Configuring the JMS Service” on page 291](#).

**MQ Client Runtime.** The client runtime part of the JMS Service is a set of libraries that support the JMS API. Any JMS clients (JMS client components, including MDBs) that run in a server instance have access to these libraries.

**MQ Administered Objects.** The built-in JMS Service uses an object store provided by application server. Each server instance has its own object store. The JMS Service stores administered objects (connection factory and destination resources) in this object store. You create these administered object resources as described in [“Managing Administered Object Resources” on page 294](#), and they are accessed by JMS clients using JNDI lookup code.

**Application Server Administration.** The Sun Java System Application Server Administration interface and command line utilities implement a limited subset of MQ administration capability. The Administration interface and command line let you configure the built-in JMS Service, create and delete physical destinations, and create and delete administered object resources needed by JMS clients to perform JMS messaging operations. These administration tools, however, do not allow (or do not facilitate) performing more sophisticated administration tasks, such as setting broker properties, tuning the MQ client runtime, modifying the MQ user repository, managing MQ security, and so forth. If you wish to perform these administration tasks for the built-in JMS Service, you have to use the administration tools installed with MQ and described in the *MQ Administration Guide*. For a comparison of MQ and Sun Java System Application Server administration capability, see the table [“Comparison of Sun Java System Message Queue and Sun Java System Application Server Administration Capability” on page 290](#).

## Disabling the Built-in JMS Service

By default, the built-in JMS Service is started up (that is, the MQ broker is started up) when the associated application server instance is started up. However, you might want to disable the automatic startup of the JMS Service when you start up a server instance, either because the server instance does not need to support JMS messaging or because the server instance uses an *external JMS Service*. (To disable the built-in JMS Service, see [“Configuring the JMS Service” on page 291](#).)



An external JMS Service is a messaging system that is not controlled from within application server. In the case of MQ—the *native* JMS provider—this means that you simply start up and manage your MQ message server independently, using MQ administration tools. JMS clients running in various server instances can still use MQ administered objects to access the MQ message server. These administered objects can either be stored in the object store associated with each application server instance, or in a separate, independent object store managed by the MQ administration tools (where the store can be shared by multiple server instances if necessary).

There are a number of scenarios in which a server instance might use an external JMS Service. The most likely is when JMS clients in different server instances need to access the same physical destinations. In that case, the server instances must all access the same message server. To achieve this, you disable the built-in JMS Service for all server instances and configure all JMS clients to perform the appropriate JNDI lookups to access the external JMS Service. In addition, you independently administer the external JMS Service (managing the message server, creating physical destinations, and creating all needed administered objects) using the external JMS Service provider's administration tools.

To configure multiple application server instances to share a single MQ broker instance:

1. Disable the JMS service on all the server instances.
2. Manage the shared MQ broker independently of any server instance, which means you must start up and shut down the broker using the administration tools for managing the external service. You must also manage physical destinations independently of the Sun Java System Application Server.
3. Configure a connection factory JMS resource in each server instance such that it points to this external MQ broker (you must set the properties appropriately).
4. Use this connection factory resource when deploying JMS applications on the Sun Java System Application Server.

It is possible to have both an external and built-in JMS Service running at the same time. JMS clients in a server instance can access whichever JMS Service they need.

One possibility that is *not* recommended is to have a number of server instances share the same built-in JMS Service (one is enabled, the rest are disabled). This is not recommended because the enabled JMS Service only runs when its associated server instance runs, making it very difficult to manage the situation.

When you disable a built-in JMS Service, you also disable your ability to perform administrative tasks associated with that built-in JMS Service. All administrative tasks needed to support an external JMS Service must be performed using the administration tools for managing that external service.

# Administration of the Built-in JMS Service

This topic focuses on administration of the built-in JMS Service. *Administration is performed on a server instance-by-server instance basis.*

Administration of the built-in JMS Service involves the following tasks:

- [Configuring the JMS Service](#)
- [Managing Physical Destinations](#)
- [Managing Administered Object Resources](#)
- [Viewing the JMS Log](#)
- [Administering the Built-in JMS Service Using the Command-Line Interface](#)

Administration can be performed using the Sun Java System Application Server’s Administration interface or command line utility. A comparison of these administration tools with MQ administration tools is shown in the table [“Comparison of Sun Java System Message Queue and Sun Java System Application Server Administration Capability.”](#)

**Table 10-1** Comparison of Sun Java System Message Queue and Sun Java System Application Server Administration Capability

Function	Sun Java System MQ Administration Tools	Sun Java System Application Server Administration interface	Sun Java System Application Server Administration Command Line
Manage MQ broker state	yes	start/stop	start/stop
Configure MQ broker	yes	no	no
Manage MQ broker user repository	yes	no	no
Multi-broker clusters	yes	no	no
Manage security	yes	no	no
Manage physical destinations	yes	create/delete	create/delete

**Table 10-1** Comparison of Sun Java System Message Queue and Sun Java System Application Server Administration Capability

Function	Sun Java System MQ Administration Tools	Sun Java System Application Server Administration interface	Sun Java System Application Server Administration Command Line
Manage durable subscriptions and transactions	yes	no	no
Manage administered object resources	yes	create/delete/configure	yes

The following sections explain how to perform JMS Service administration tasks using the Sun Java System Application Server Administration interface.

## Configuring the JMS Service

At installation time a number of JMS Service properties are set for the built-in JMS Service. You can change the default values of these properties by configuring the JMS Service.

JMS Service properties are described in the [Table 10-2](#).

**Table 10-2** JMS Service Properties

Property	Description	Default Value
Log level	The level of logging information you want written to the Sun Java System Application Server log file. For more information, see <a href="#">Chapter 5, "Using Logging."</a>	DEBUG_HIGH
Port	<p>The primary port number of the broker instance providing built-in JMS Service. By default, the built-in JMS Service uses the default primary port number. However, if that port conflicts with other software, or if you are starting up more than one Application Server instance, you need to specify a unique primary port number for each.</p> <p>Note that it is possible to have a port conflict if the port number assigned to the JMS Service at installation time later is used by another service. In this case, you have to give the JMS Service a different port number.</p>	7676

**Table 10-2** JMS Service Properties (*Continued*)

Property	Description	Default Value
Administrator's username/password	The username/password needed to perform broker administration tasks, such as managing physical destinations (see <a href="#">“Managing Physical Destinations” on page 293</a> ). It is recommended that you modify the administrator username and password to be something other than the default. To do this, you need to first make the appropriate entries in the broker's user repository (as described in the <i>Sun Java System Message Queue Administrator's Guide</i> ) and then enter the corresponding values here for the administrator's username and password.	admin/admin
Start Timeout	Specifies the time in seconds that the server instance waits for the JMS Service to start up. If this timeout is exceeded, the server instance startup is aborted.	60
Start Arguments	Specifies any arguments that will be used in starting up the JMS Service. The startup arguments for the <code>imqbroker</code> command, and how to specify them, can be found in the <i>Sun Java System Message Queue Administrator's Guide</i> . (The <code>-name</code> and <code>-port</code> arguments, if supplied, are ignored.)	
Startup Enabled	Specifies whether the built-in JMS Service is started up when the server instance starts up. If you are not supporting JMS messaging or are using an external JMS message service, set this property to FALSE.	TRUE

The built-in JMS Service can be configured before starting up its corresponding server instance. If the server instance is already running, however, configuration changes take effect only after the server instance is stopped and subsequently restarted.

To configure the built-in JMS Service:

1. Open the Administration interface.
2. Open a server instance in the left pane.
3. Open the JMS folder.
4. Select the Service link.
5. Modify any of the properties you wish (see the table [“JMS Service Properties” on page 291](#)).
6. Click the Save button.

The JMS Service screen is refreshed.

## Managing Physical Destinations

In JMS messaging, a JMS producer sends messages to a physical destination on a message service from which they are dispatched to a JMS consumer.

For the built-in JMS Service, you can create these physical destinations explicitly or they can be created automatically by the JMS Service (MQ broker) upon receipt of a message. In general, you have more control over the messaging system and its resources by explicitly creating the physical destinations needed by messaging applications. When these destinations are no longer needed, you can delete them.

In order to create or delete physical destinations for a built-in JMS Service, the JMS Service must be running and the administrator username and password (specified when you configure the built-in JMS Service) must correspond to valid entries in the broker's user repository (see the table [“JMS Service Properties” on page 291](#)).

Using the Administration interface, you can perform the following management tasks for physical destinations on a built-in JMS Service:

- [Create a Physical Destination](#)
- [List Physical Destinations](#)
- [Delete a Physical Destination](#)

### Create a Physical Destination

To create a queue or topic destination:

1. Open the Administration interface.
2. Open a server instance in the left pane.
3. Open the JMS folder.
4. Open the Service link.
5. Select the Physical Destinations link

The Physical Destinations screen is displayed in the right pane.

6. Click the New button.

The Physical Destinations: New screen is displayed in the right pane.

7. Enter the name of the physical destination.
8. Select queue or topic from the Type pull down.

9. Click the OK button.

The right pane is refreshed and displays the new queue or topic destination in the list of existing queue and topic destinations.

## List Physical Destinations

To list existing queue and topic destinations:

1. Open the Administration interface.
2. Open a server instance in the left pane.
3. Open the JMS folder.
4. Open the Service link.
5. Select the Physical Destinations link

The right pane displays the current physical destinations.

## Delete a Physical Destination

You can delete queue or topic destinations as needed.

To delete a physical destination:

1. List the existing destinations (see [“List Physical Destinations” on page 294](#)).
2. Click within the Select box for each destination you wish to delete.
3. Click the Delete button to remove the selected destinations.

The list refreshes, showing the remaining destinations.

# Managing Administered Object Resources

MQ administered objects, regarded by Sun Java System Application Server as *JMS Resources*, are used by JMS clients to access a JMS Service (either built-in or external).

Two administered object resources—connection factories and destinations—are used by J2EE components to obtain connections to a JMS Service, and then to deliver messages to and from physical destinations on the Service (see [“MQ Administered Objects” on page 285](#)).

Because the creation of administered object resources does not involve the JMS Service directly, you do not need to enable the JMS Service, and you do not need a valid username and password (see the table [“JMS Service Properties” on page 291](#)), to create administered object resources for a server instance.

## Administered Object Attributes

To support JMS messaging, you must create the administered object resources required by all JMS clients running in a server instance. At a minimum, you need to specify a JNDI lookup name for each administered object resource, its type (connection factory, queue, or topic), a description (optional) and whether or not the resource is enabled. Other attributes are discussed in the following sections.

### *Destination (queue or topic)*

In the case of queue or topic administered objects, you also need to specify the name of the corresponding physical destination.

### *Connection Factory*

Connection factory administered objects have attributes as listed in Table 10-3. The attribute you are primarily concerned with is `imqAddressList`, which is used to specify the broker to which the client will establish a connection. The section, [“Create a ConnectionFactory Administered Object” on page 298](#) explains how you specify a attributes when you add a connection factory administered object to your object store.

**Table 10-3** Connection Factory Administered Object Attributes

Attribute/property name	Type	Default Value
<code>imqAckOnAcknowledge</code>	String	No value
<code>imqAckOnProduce</code>	String	No value
<code>imqAckTimeout</code>	String	0 millisecs
<code>imqAddressList</code>	String	No value
<code>imqAddressListIterations</code>	Integer	1
<code>imqAddressListBehavior</code>	String	PRIORITY
<code>imqBrokerHostName</code> (MQ 3.0)	String	localhost
<code>imqBrokerHostPort</code> (MQ 3.0)	Integer	7676
<code>imqBrokerServicePort</code> (MQ 3.0)	Integer	0
<code>imqConfiguredClientID</code>	String	No value
<code>imqConnectionFlowCount</code>	Integer	100
<code>imqConnectionFlowLimit</code>	Integer	1000

**Table 10-3** Connection Factory Administered Object Attributes

Attribute/property name	Type	Default Value
imqConnectionFlowLimitEnabled	Boolean	false
imqConnectionType (MQ 3.0)	String	TCP
imqConnectionURL (MQ 3.0)	String	<a href="http://localhost/imq/tunnel">http://localhost/imq/tunnel</a>
imqConsumerFlowLimit	Integer	1000
imqConsumerFlowThreshold	Integer	50
imqDefaultPassword	String	guest
imqDefaultUsername	String	guest
imqDisableSetClientID	Boolean	false
imqJMSDeliveryMode	Integer	2 (persistent)
imqJMSExpiration	Long	0 (does not expire)
imqJMSPriority	Integer	4 (normal)
imqLoadMaxToServerSession	Boolean	true
imqOverrideJMSDeliveryMode	Boolean	false
imqOverrideJMSExpiration	Boolean	false
imqOverrideJMSHeadersToTemporaryDestinations	Boolean	false
imqOverrideJMSPriority	Boolean	false
imqQueueBrowserMaxMessagesPerRetrieve	Integer	1000
imqQueueBrowserRetrieveTimeout	Long	60,000 (milliseconds)
imqReconnectAttempts	Integer	0
imqReconnectEnabled	Boolean	false
imqReconnectInterval	Long	3000 (milliseconds)
imqSetJMSXAppID	Boolean	false
imqSetJMSXConsumerTXID	Boolean	false
imqSetJMSXProducerTXID	Boolean	false
imqSetJMSXRcvTimestamp	Boolean	false
imqSetJMSXUserID	Boolean	false
imqSSLIsHostTrusted (MQ 3.0)	Boolean	true



For more descriptions of connection factory attributes and information on how they are used, see the *Sun Java System Message Queue 3.5 Administration Guide* and the JavaDoc API documentation for the following MQ class:  
`com.sun.messaging.ConnectionConfiguration`.

However, if you disable the JMS Service for a particular server instance, then any JMS clients supported by that server instance must use an external JMS Service. When you create a connection factory to be used to create connections to that external JMS Service, you need to set attributes that specify the hostname, port number of the appropriate broker instance, address list and so on.

Connection factory administered objects have additional attributes used to tune the server instance's MQ client runtime. These are documented in the MQ *Developer's Guide*.

Connection factory administered objects created using the Administration interface support distributed transaction managers.

## Administered Object Resource Management Tasks

From the Administration interface, you can perform the following management tasks for administered object resources:

- [Create a Queue or Topic Administered Object \(Destination Resource\)](#)
- [Create a ConnectionFactory Administered Object](#)
- [List Administered Object Resources](#)
- [Delete an Administered Object Resource](#)

### *Create a Queue or Topic Administered Object (Destination Resource)*

To create a queue or topic administered object:

1. Open the Administration interface.
2. Open a server instance in the left pane.
3. Open the JMS folder.
4. Select the Destination Resources link.

The Destination Resources screen is displayed in the right pane.

5. Click on the New button.

The Destination Resources: New screen is displayed.

6. Enter the JNDI lookup name associated with this destination administered object.

7. Select the “queue” or “topic” object type from the pull down list.
8. Click the OK button.

The right pane re-displays the Destination Resource: New screen.

In addition, you must specify the destination name for the object by specifying the `imqDestinationName` property. The value of this property should match the physical destination’s name.

To change the value of this property:

1. Open the Administration interface.
2. Open a server instance in the left pane.
3. Open the JMS folder.
4. Open the Destination Resources folder.
5. Select the Destination Resource you want to edit.

The right pane displays the Destination Resource screen.

6. In the right pane, click Properties.

The Edit Properties screen appears.

7. In the Name field, enter `imqDestinationName`.
8. In the Value field, enter the physical destination’s name.
9. Click OK.

The right pane redisplay the Destination Resources screen.

### *Create a ConnectionFactory Administered Object*

To create a queue connection factory or topic connection factory administered object:

1. Open the Administration interface.
2. Open a server instance in the left pane.
3. Open the JMS folder.
4. Select the Connection Factories link.

The Connection Factory Resources screen is displayed in the right pane.

5. Click on the New button.

The Connection Factory Resources: New screen is displayed.

6. Enter the JNDI lookup name associated with this connection factory administered object.
7. Select the connection factory object type from the pull down list.
8. Click the OK button.

The right pane re-displays the Connection Factory Resources screen with the newly created connection factory object in the list.

If this connection factory creates connections to a broker other than that of the built-in JMS Service, you must set values for other properties.

To change the value of these properties:

1. Open the Administration interface.
2. Open a server instance in the left pane.
3. Open the JMS folder.
4. Open the Connection Factories folder
5. Select the Connection Factory resource you want to edit.

The right pane displays the Connection Factory screen.

6. In the right pane, click Properties.

The Edit Properties screen appears.

7. In the Name field, enter a property.
8. In the Value field, enter a value for the property.
9. Enter other property and the respective values.
10. Click OK.

The right pane redisplay the Connection Factory screen.

### *List Administered Object Resources*

To list existing administered objects:

1. Open the Administration interface.
2. Open a server instance in the left pane.
3. Open the JMS folder.

4. Select the Destination Resources or Connection Factory Resources link

The right pane displays the current destination or connection factory administered objects.

### *Delete an Administered Object Resource*

To delete an administered object resource:

1. List the existing administered object resources (see [“List Administered Object Resources” on page 299](#)).

The right pane displays the existing administered object resources.

2. Click within the Select box for each object you wish to delete.
3. Click the Delete button to remove each selected object.

The screen refreshes the list, showing the remaining administered object resources.

## Viewing the JMS Log

To troubleshoot your JMS Service, in addition to the Sun Java System Application Server log files, you can also use the JMS log file. All the messages from the MQ broker are logged to the broker log files, while the Sun Java System Application Server log files contain the messages which the MDB or the MDB container sends.

The location of the JMS log file varies by platform and installation type.

For the file-based installation, the log file default location is:

*install\_root*/imq/instances/*instance*/log/log.txt

For Solaris package-based installations, the log file default location is:

/var/imq/instances/*instance*/log/log.txt

For Linux packaged-based installations, the log file default location is:

/opt/imq/var/instances/*instance*/log/log.txt

For Windows installations, the log file default location is:

*install\_root*/imq/instances/*instance*/log/log.txt

For more information on using the JMS log files, see the *Sun Java System Message Queue Administration Guide*.

## Administering the Built-in JMS Service Using the Command-Line Interface

The Sun Java System Application Server has a command-line utility, `asadmin`, which you can use to perform all the same tasks as you can perform with the Administration interface.

Use the following `asadmin` commands to configure and administer the built-in JMS Service.

**Table 10-4** `asadmin` Commands for Administering the Built-in JMS Service

Command	Use
<code>add-resources</code>	Adds one or more resources of type <code>jdbc</code> , <code>jms</code> , or <code>javamail</code> .
<code>create-jmsdest</code>	Creates a JMS physical destination.
<code>create-jms-resource</code>	Creates a JMS resource.
<code>delete-jmsdest</code>	Deletes a JMS physical destination.
<code>delete-jms-resource</code>	Deletes a JMS resource.
<code>jms-ping</code>	Pings the JMS provider to see if it is running.
<code>list-jmsdest</code>	Lists JMS physical destinations for a server instance.
<code>list-jms-resources</code>	Lists JMS resources for a server instance.
<code>get</code> and <code>set jms-service</code>	Gets and sets attributes for the JMS service.
<code>get</code> and <code>set jms-resource</code>	Gets and sets attributes for a JMS resource.

For information on the syntax of these commands, see the online help for `asadmin`. For more information on `asadmin`, as well as a list of attributes for `jms-service` and `jms-resource`, see [Appendix A, “Using the Command Line Interface.”](#)



# Configuring the Server For Corba/IIOP Clients

This chapter explains how to configure support for CORBA/IIOP clients, using the RMI/IIOP protocol within the Application Server environment.

This chapter includes the following topics:

- [About Support for CORBA/IIOP Clients](#)
- [Configuring the ORB](#)

## About Support for CORBA/IIOP Clients

The J2EE platform provides indirect support for various types of clients, different hardware platforms, and a multitude of software applications through its interoperability requirements. As a J2EE-compliant product, Application Server supports a standard set of protocols and formats that ensure interoperability.

The CORBA (Common Object Request Broker Architecture) model is based on clients requesting services from distributed objects or servers through a well-defined interface, by issuing requests to the objects in the form of remote method requests. A remote method request carries information about an operation that needs to be performed including the object name (called an object reference) of the service provider and the actual parameters, if there are any. CORBA automatically handles a lot of network programming tasks such as object registration, object location, object activation, request de-multiplexing, error-handling, marshalling and operation dispatching.

The following topics are covered in this section:

- [About Interoperability](#)
- [About the ORB](#)
- [About the RMI/IIOP Functionality](#)
- [About the Authentication Process](#)

## About Interoperability

Interoperability essentially means the ability of an enterprise environment to bring together applications written in various languages. One or more of these existing applications may be running on a personal computer platform, while others may be running on UNIX. In addition, these enterprise environments may also be supporting standalone Java technology based applications that are not directly supported by the J2EE platform.

J2EE is mandated to provide support for CORBA IIOP (Internet Inter-Orb Protocol) protocol. CORBA defines a model that specifies interoperability between distributed objects on a network in a way that is transparent to the user. CORBA achieves this by defining ways for specifying the externally visible characteristics of a distributed object in a way that is implementation-independent.

## About the ORB

Object Request Broker (ORB for short) is the central component of CORBA. The ORB provides the required infrastructure to identify and locate objects, handle connection management, deliver data and request communication.

One CORBA object never talks directly with another. Instead, the object makes requests through a remote stub to the ORB running on the local machine. The local ORB then passes the request to an ORB on the other machine using Internet Inter-Orb Protocol (IIOP for short). The remote ORB then locates the appropriate object (servant) processes the request and returns the results. IIOP can be used as a Remote Method Invocation (RMI for short) protocol by JAVA applications or objects, using the RMI-IIOP technology.

The ORB bundled with Sun Java System Application Server provides support for load balancing RMI/IIOP application requests. For information on how to enable load balancing in your client applications, see [Chapter 18, “Configuring RMI/IIOP Load Balancing and Failover \(Enterprise Edition\).”](#)



## About the RMI/IIOP Functionality

CORBA specifies the ORB which allows applications to communicate with each other regardless of location. This interoperability is delivered through IIOP, and is typically found in an Intranet setting. Some of the functionalities achieved by RMI over IIOP are as follows:

- Interoperability with objects written in other languages.
- Ability to propagate transaction and security context.
- Plug-and-play environment for ORB services.
- Interoperability with EJBs
- Use of the COSNaming service, an IIOP-based naming service. The EJB interoperability protocol requires the use of the COSNaming to look up EJB objects using the Java Naming Directory Interface (JNDI for short) API.

The JAVA ORB that comes bundled with Application Server supports the following functionalities:

- Conformance level 0 of CSIV2 (Common Secure Interoperability version 2).
- Fully compliant COSNaming service implements the IDL interfaces and aid the EJB container to publish EJBHome references.
- IIOP/GIOP Ver 1.2. CORBA specifies the ORB which allows applications to communicate with each other regardless of location. This interoperability is delivered through IIOP.

---

<b>NOTE</b>	Failover and load balancing of remote references of EJB and Name Server objects along the RMI/IIOP path is available in the Enterprise Edition of Sun Java System Application Server. For more information on how to configure RMI/IIOP load balancing, see <a href="#">Chapter 18, “Configuring RMI/IIOP Load Balancing and Failover (Enterprise Edition).”</a>
-------------	--

---

## About the Authentication Process

Authentication is the process of confirming an identity. In the context of network interactions, authentication is the confident identification of one party by another party. Certificates are one way of supporting authentication.

The following two kinds of authentication are applicable:

**Server Authentication.** Server authentication refers to the confident identification of a server by a client; that is, identification of the organization assumed to be responsible for the server at a particular network address.

**Client Authentication.** Client authentication refers to the confident identification of a client by a server; that is, identification of the person assumed to be using the client software.

Clients can have multiple certificates, much like a person might have several different pieces of identification.

## Configuring the ORB

You can configure multiple IIOP-listeners for each instance of Application Server. By default, one IIOP listener is configured. You can configure the IIOP listener properties for your ORB and add additional listeners.

You can also enable monitoring for the ORB, specify the log level at which messages will be logged, specify thread pool settings, and configure IIOP listener ports and SSL configuration for the IIOP path. In this section, we will discuss how to configure ORB support for an instance of Application Server.

The following topics are included in this section:

- [To Perform General ORB Configuration](#)
- [To Configure IIOP Listener For the ORB](#)

### To Perform General ORB Configuration

Using the Administration interface, you can enable monitoring, set log levels, and configure pool settings for the thread pool. To perform general ORB configuration, perform the following tasks:

1. In the left pane of the Administration interface, expand the Application Server instance for which you want to configure ORB settings.
2. Click the ORB tab. The General ORB Configuration page is displayed.
3. In the General section of this window, you can enable monitoring, and set log levels for your ORB.

- a. To enable monitoring for the ORB, mark the Monitoring Enabled checkbox.
- b. Choose the log level you want, from the Log Level drop-down list. The default log level for the server is typically set to INFO. The default level for the ORB is to use the default for the server. The log level will therefore display Default (INFO), in the drop-down list.

Log levels are provided to record messages of a range of severity, from FINEST to FATAL. Setting a log level allows you to select what granularity of messages are displayed in the log. A granularity of WARNING will display WARNING, ALERT, SEVERE and FATAL messages. Normally you would need to set the granularity at the server-wide level, but you can use this setting to control the messages displayed from the Application Server ORB.

- 4. In the Thread Pool section of this window, you can specify the pool settings for the request threads used by the ORB.

Request threads handle user requests for application components. When Application Server receives a request, it assigns the request to a free thread from the thread pool. The thread executes the client's requests and returns results. For example, if the request needs to use a system resource that is currently busy, the thread waits until that resource is free before allowing the request to use that resource.

You can specify the minimum and maximum number of threads that are reserved for requests from applications. The thread pool is dynamically adjusted between these two values. The minimum thread-pool size you specify signals the ORB to allocate at least that many threads in reserve for application requests. That number is increased up to the maximum thread-pool size that you specify.

Increasing the number of threads available to a process allows the process to respond to more application requests simultaneously.

- a. In the Steady Pool Size field, specify the minimum number of threads in the pool. The pool will also shrink to this number after threads are idle for the period specified in the Idle Timeout (secs) field.
- b. In the Max Pool Size field, specify the maximum number of threads to which the thread pool can grow.
- c. In the Idle Timeout (secs) field, specify the timeout for the idle threads in the threadpool to be cleaned up.

5. In the Advanced section of this window, you can configure advanced options for your ORB, as follows:
  - a. In the Message Fragment Size field, specify the maximum GIOP 1.2 message size, in order to support fragmentation. The default fragment size is 1024.
  - b. In the Total Connections field, specify the maximum number of incoming remote IIOP connections allowed by the ORB server process.
6. Click Save to save your settings. If you want to revert to your previous settings without saving the recent changes, click Revert.

## To Configure IIOP Listener For the ORB

Each new instance of Application Server comes with a default ORB configuration, which includes a pre-configured IIOP listener. The IIOP listener is a listen socket that listens on a specified port and accepts incoming connections from CORBA based client application. You can configure any number of IIOP listeners for a single instance of Application Server.

To create a new IIOP listener or to configure IIOP listener properties, perform the following tasks:

1. In the left pane of the Administration interface, expand the Application Server instance for which you want configure ORB properties.
2. Click ORB, and open the IIOP Listener tab under it. You will see a list of all the IIOP Listeners that have been configured for that specific instance of Application Server.
3. To create a new IIOP Listener, click New (if you are editing an existing IIOP listener, just open the listener and perform tasks listed in the following steps). When you click New, or when you open an existing IIOP listener, you will see the Creating a New IIOP Listener page.
4. You can configure general parameters for your IIOP listener, as follows:
  - a. In the Id text field, provide a name to identify the listener. You can use any identifier, such as *ORB\_Listener1*, *ORB\_Listener2*, etc.
  - b. In the Address text field, type the address of the machine on which you have installed Application Server. You can either specify the machine address in the *machinename.domainname* format, as indicated in the given example, or you can provide the IP address of the machine.

- c. In the Port text field, type a unique port number for the new IIOP Listener. The default IIOP listener comes with a default port number. You can change this port number. However, before changing the port number, please ensure that the new port number that you specify is not being used by any other existing software application or process.
  - d. To enable the listener, mark the Listener Enabled checkbox.
- 5. In the SSL/TLS Settings section on this page, you can set security for the IIOP listener. Check the appropriate boxes associated with the Secure Sockets Layer (SSL) and Transport Layer Security (TLS), including all the ciphers. You can select either SSL2 or SSL3/TLS sockets. You can configure the SSL/TLS settings for your listener, as follows:
  - a. In the Certificate Nickname field, provide the nickname of the certificate that the server presents to the client during SSL handshake. You must have previously installed a certificate to see its nickname in this list.
  - b. Mark the SSL2 Enabled field, to enable SSL2 security option for the listener path.
  - c. Select the SSL2 ciphers that you want to use for the SSL2 security. Mark the checkboxes against the required ciphers. Unless you have a compelling reason for not using a specific cipher suite, you should allow them all.
  - d. Mark the SSL3 Enabled field, to enable SSL3 security option for the listener path.
  - e. Mark the TLS Enabled field, to enable TLS. TLS must also be enabled on the browser seeking access to your server. Check both TLS and SSL3 for Netscape Navigator 6.0.
  - f. Mark the TLS Rollback Enabled field. In order to enable TLS Rollback, you need to enable TLS first. Also ensure that SSL3 and SSL2 are disabled, when you enable this option. Use the TLS Rollback option for Microsoft Internet Explorer 5.0 and 5.5.
  - g. Select the SSL3/TLS ciphers that you want to use for SSL3 and TLS. Select these only if you have enabled SSL3 or TLS. Unless you have a compelling reason for not using a specific cipher suite, you should allow them all.
  - h. Mark the Client Authentication Enabled checkbox to indicate whether the ORB listener port for SSL IIOP connections with client authentication is enabled or not. Client authentication is the process of authenticating client certificates by cryptographically verifying the certificate signature and the certificate chain leading to the CA on the trust CA list.
- 6. Click OK to save the IIOP listener settings.

---

**NOTE**

- When you install Application Server, an IIOP listener is created for the default server instance. The default port number for the default IIOP listener port is 3700.
  - Please note that each IIOP listener must bear a different port number. Also note that the machine address that you provide in the Address text field must be the address of the machine on which Application Server is installed.
  - For more information about SSL settings for the listener path, and other details of security for Application Server, see the *Sun Java System Application Server Administration Guide to Security*.
-

# Configuring the Web Server Plug-in

This chapter explains how Sun Java™ System Application Server Standard Edition 7 processes HyperText Transfer Protocol (HTTP) requests, and how to configure and use the web server plug-in with the Application Server. This chapter also explains how to configure and use the web server plug-in with Microsoft IIS, and Apache web server.

---

**NOTE** The procedures documented in this chapter apply only to Sun Java System Application Server 7, Standard Edition web server plug-in. This plug-in does not provide HTTP session failover capabilities.

HTTP load balancing and failover is available in the enterprise edition and is described in [Chapter 17, “Configuring HTTP Load Balancing and Failover \(Enterprise Edition\).”](#)

---

This chapter includes the following topics:

- [About the Web Server Plug-in](#)
- [Handling Client Requests](#)
- [Web Server Plug-in Configuration](#)
- [Web Server Plug-in SAF Reference](#)
- [Using The Web Server Plug-in](#)

## About the Web Server Plug-in

The web server plug-in is an HTTP reverse proxy plug-in that allows you to instruct a Sun Java System Web Server or the Application Server to forward certain HTTP requests to another server. For example, you can configure a web server connected to the Internet to forward requests for specific web applications to an Application Server located behind a corporate firewall.

Within Sun Java System Application Server, the web server plug-in allows one server instance to forward an HTTP (web) request to another server instance.

The web server plug-in performs the following functions:

- Re-uses connections from the proxy server, whenever possible. This eradicates the necessity of opening new connections to process incoming requests.
- The web server plug-in starts streaming requests and responses as it starts receiving them. In other words, the plug-in does not wait till the request or response is collected fully before forwarding it to the remote server.
- The web server plug-in maintains multiple outbound HTTP connections to the same remote server, as appropriate. Connections formed for requests that are forwarded by the web server plug-in are called outbound HTTP connections.

To understand how the web server plug-in works, it is necessary to understand the basics of HTTP requests, and specifically the method used by Sun Java System Application Server to process HTTP requests.

## Handling Client Requests

In this section, we will discuss HTTP basics in and also look at how Sun Java System Application Server handles requests. This section covers the following topics:

- [HTTP Basics](#)
- [Steps in the Request Handling Process](#)

### HTTP Basics

As a quick summary, the HTTP/1.1 protocol works as follows:

- The client (usually a browser) opens a connection to the server and sends a request



- The server processes the request, generates a response, and closes the connection if it finds a `Connection: Close` header.

The request consists of a line indicating a method such as `GET` or `POST`, a Universal Resource Identifier (URI) indicating which resource is being requested, and an HTTP protocol version separated by spaces.

This is normally followed by a number of headers, a blank line indicating the end of the headers, and sometimes body data. Headers may provide various information about the request or the client body data. Headers are typically only sent for `POST` and `PUT` methods.

The example request shown below would be sent by a browser to request the server `foo.com` to send back the resource in `/index.html`. In this example, no body data is sent because the method is `GET` (the point of the request is to get some data, not to send it.)

```
GET /index.html HTTP/1.0
User-agent: Mozilla
Accept: text/html, text/plain, image/jpeg, image/gif, */*
Host: foo.com
```

The server receives the request and processes it. It handles each request individually, although it may process many requests simultaneously. Each request is broken down into a series of steps that together make up the request handling process.

The server generates a response which includes the HTTP protocol version, HTTP status code, and a reason phrase separated by spaces. This is normally followed by a number of headers. The end of the headers is indicated by a blank line. The body data of the response follows. A typical HTTP response might look like this:

```
HTTP/1.0 200 OK
Server: Standard/7.0
Content-type: text/html
Content-length: 83

<HTML>
<HEAD><TITLE>Hello World</Title></HEAD>
<BODY>Hello World</BODY>
</HTML>
```

The status code and reason phrase tell the client how the server handled the request. Normally the status code 200 is returned indicating that the request was handled successfully and the body data contains the requested item. Other result codes indicate redirection to another server or the browser's cache, or various types of HTTP errors such as "404 Not Found."

## Steps in the Request Handling Process

When Sun Java System Application Server first starts up, it performs certain initialization tasks, and then waits for an HTTP request from a client (such as a browser). When it receives a request, it first selects a virtual server.

After the virtual server is selected, the virtual server's `obj.conf` file specifies how the request is handled with the following steps:

1. **AuthTrans** (authorization translation)

Verify any authorization information (such as name and password) sent in the request.

2. **NameTrans** (name translation)

Translate the logical URI into a local file system path.

3. **PathCheck** (path checking)

Check the local file system path for validity and check that the requestor has access privileges to the requested resource on the file system.

4. **ObjectType** (object typing)

Determine the MIME-type (Multi-purpose Internet Mail Encoding) of the requested resource (for example, `text/html`, `image/gif`, and so on).

5. **Service** (generate the response)

Generate and return the response to the client.

6. **AddLog** (adding log entries)

Add entries to log file(s).

7. **Error** (service)

This step is executed only if an error occurs in the previous steps. If an error occurs, the server logs an error message and aborts the process.

# Web Server Plug-in Configuration

The configuration and behavior of the web server plug-in are determined by a set of configuration files. Application Server looks at the configuration defined in these files each time it processes a request from a client. The configuration files are named `obj.conf` and `init.conf`. The `obj.conf` file is prefixed with the name of the virtual server, for example `server1-obj.conf`. For more information, see [“obj.conf File” on page 367](#).

Each instance of Sun Java System Application Server has its own `init.conf` file, to which the server refers at startup.

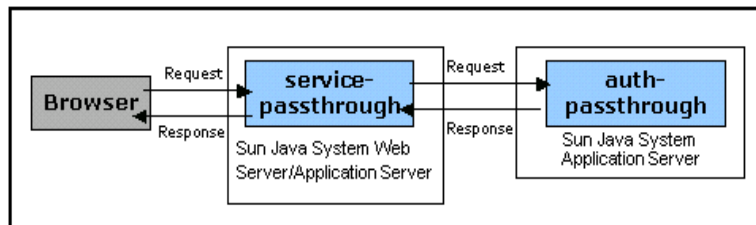
As discussed in the preceding topic, the `obj.conf` configuration file contains a series of instructions (directives) that tell Application Server what to do at each stage in the client request and response process. Each directive invokes a Server Application Function (SAF).

The `obj.conf` file is essential to the operation of the Application Server. When you make changes to the server through the Administration interface, the system automatically updates `obj.conf`.

The `init.conf` configuration file sets values of variables that configure the server during initialization. The server executes the configuration parameters specified in this file, during server start up. See the *Sun Java System Application Server Administrator's Configuration File Reference* for more information.

The following figure illustrates the relationship between the web browser, a front-end web server, a backend Application Server, and the web server plug-in's `service-passthrough` and `auth-passthrough` SAFs:

**Figure 12-1** Relationship Between Web Browser, Web Server, Application Server, and Web Server Plug-in SAFs



# Web Server Plug-in SAF Reference

This section discusses the function and behavior of the following Server Application Functions (SAF):

- [init-passthrough](#)
- [auth-passthrough](#)
- [service-passthrough](#)
- [check-passthrough](#)

## init-passthrough

The `init-passthrough` function initializes the web server plug-in. This function must be called before the web server plug-in can be used.

Example:

```
Init fn="load-modules" shlib="c:/plugins/passthrough.dll"  
funcs="init-passthrough,auth-passthrough,check-passthrough,service-passtr  
ough" NativeThread="no"  
  
Init fn="init-passthrough"
```

## auth-passthrough

The `auth-passthrough` SAF is applicable in `AuthTrans-class` directives.

The `auth-passthrough` function inspects the incoming HTTP (web) request for client information encoded by a `service-passthrough` function running on an intermediate server. The client information includes:

- IP address request originated from
- SSL key size used by originating client
- SSL client certificate presented by originating client

When `auth-passthrough` detects encoded client information, it treats the request as a direct request from the originating client and not as a request forwarded by an intermediate server running `service-passthrough`.

This is useful in two-tier deployment scenarios where:

- a Sun Java System Application Server instance is hidden by a second firewall behind the corporate firewall
- no client connections are permitted directly to the S1AS instance.

In such network architectures, a client always connects to the front-end web server which is running the proxy plug-in. This web server is the one that forwards requests to Application Server. This indicates that Application Server can only receive requests from the proxy host (in this case, the web server), and never directly from client hosts. This means that if applications, deployed on a Application Server instance that is behind two firewalls, query for client information such as the client's IP address, the applications will get the proxy host IP (since that is the actual originating host for the relayed request). The `auth-passthrough` SAF can be used to modify this behavior so that the remote (proxied) client information is presented instead.

Since `auth-passthrough` makes it possible to override information that may be used for authentication (for example, the IP address from which the request originated) it is important that only trusted clients or servers be allowed to connect to a server running `auth-passthrough`. As a precautionary measure, it is recommended that only servers behind the corporate firewall should run `auth-passthrough`. A server that is accessible through the Internet should not run the `auth-passthrough` SAF. The `auth-passthrough` SAF should be used only if the relevant information about the originating client is required.

Note that in the scenario described above, SSL client authentication could be turned 'on' only for web server and turned 'off' always for Application Server. For the configuration to work properly, the Application Server's `obj.conf` and `init.conf` configuration files must be suitably modified, as described in [“Changes to Sun Java System Application Server” on page 321](#).

## service-passthrough

The `service-passthrough` SAF is applicable in Service-class directives.

The `service-passthrough` SAF forwards a request from one server to another server for processing. The `service-passthrough` SAF can be configured to use SSL or non SSL (HTTPS or HTTP) connections to the remote server, independent of the type of connection through which the original request was received. The `service-passthrough` SAF encodes information about the originating client that may be decoded by an `auth-passthrough` function running on the remote server.

A `service-passthrough` directive is typically used in combination with other directives in the `obj.conf` configuration file as follows:

```
<Object name="passthrough">
  ObjectType fn="force-type" type="magnus-internal/passthrough"
  Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</Object>

<Object name="default">
  ....
  NameTrans fn="assign-name" from="( /webapp1 | /webapp1/* )" name="passthrough"
  ...
</Object>
```

If the backend Application Server is down, the user will be shown the local HTML file `badgateway.html` instead. In case the server running the `service-passthrough` SAF needs to serve files to which it has access, and forward only rejected requests to the backend Application Servers, the `ObjectType` line would be changed to:

```
ObjectType fn="check-passthrough" type="magnus-internal/passthrough"
```

## check-passthrough

The `check-passthrough` SAF is Applicable in `ObjectType-class` directives.

The `check-passthrough` function checks to see if the requested resource (for example, an HTML document, or a GIF image) is available on the local server. If the requested resource does not exist locally, the `check-passthrough` SAF sets the type to indicate that the request should be passed to another server to be processed by the `service-passthrough` SAF.

### Parameters:

`type` - (optional) type to set when the request resource does not exist. The default is "magnus-internal/passthrough".

### Example

```
ObjectType fn="check-passthrough"
```

# Using The Web Server Plug-in

To use the web server plug-in as a proxy to Sun Java System Application Server, you must make changes to the configuration files of both the Application Server and Web Server. Follow the procedures listed in this section to configure and use the Reverse Proxy Plug-in.

- [Configuring Sun Java System Web Server](#)
- [Configuring Application Server as a Proxy](#)
- [Configuring Microsoft IIS](#)
- [Configuring Apache Web Server](#)

## Configuring Sun Java System Web Server

To configure Sun Java System Web Server as a proxy, you need to make changes to both Web Server and Application Server configuration files. These changes are described in the following sections:

- [Changes to Sun Java System Web Server](#)
- [Changes to Sun Java System Application Server](#)

## Changes to Sun Java System Web Server

Take backups of critical configuration files, such as `magnus.conf` and `obj.conf`, before making changes to these files.

1. Create a directory in the web server installation area that will contain the passthrough plug-in. For example:

```
cd /webserver_install_dir/plugins
```

```
mkdir -p passthrough/bin
```

2. Copy the passthrough plug-in from Sun Java System Application Server installation to this new, web server directory. For example:

```
cd appserver_install_dir/lib/webserver-plugin/platform/web_server/
```

```
cp libpassthrough.so webserver_install_dir/plugins/passthrough/bin
```

On Windows, copy the `passthrough.dll` file.

3. Create a directory, `lbplugin/resource`, under `web_server_install_dir/plugins`.
4. Copy all `*.res` files from `appserver_install_dir/lib/webserver-plugin/platform/web_server` to `web_server_install_dir/plugins/lbplugin/resource`.
5. Edit the `magnus.conf` file, found under `webserver_install_dir/https-host.domain/config`, and append the following lines:

```
Init fn="load-modules"
shlib="webserver_install_dir/plugins/passthrough/bin/libpassthrough.so"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-passthrough"
NativeThread="no"
Init fn="init-passthrough"
```

6. Edit the `obj.conf` file, found under `webserver_install_dir/https-host.domain/config`, and add the `NameTrans` directive as shown:

```
<Object name="default">
NameTrans fn="assign-name" from="/*" name="passthrough"
...
</Object>
```

The `from="/*"` URI is the context root of a web application deployed on the remote servers, and `passthrough` corresponds to the name of the `<Object>` in `obj.conf`.

For example:

```
<Object name="default">
NameTrans fn="assign-name" from="(/webappl|/webappl/*)" name="passthrough"
...
</Object>
```

7. Add the following lines in the `obj.conf` file:

```
<Object name="passthrough">
ObjectType fn="force-type" type="magnus-internal/passthrough"
PathCheck fn="deny-existence" path="*/WEB-INF/*"
Service type="magnus-internal/passthrough" fn="service-passthrough"
servers="http://servername:port"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</Object>
```

`servername` is the hostname or IP address of machine on which Sun Java System Application Server is installed, and `port` is the number of the port on which it listens (this value is typically set to 80).



8. Include the path of plug-in file to the `PATH` variable.
9. Restart the Sun Java System Web Server instance.

Next, you must configure Sun Java System Application Server to receive SSL requests from the web server plug-in.

---

**NOTE** Ensure that the `KeepAliveTimeout` value is the same in both the web server's `magnus.conf` file and the Application Server's `init.conf` file. If the values are different, the connection might get closed prematurely, either by the web server or Application Server, and you will see a 502 error in the browser.

---

## Changes to Sun Java System Application Server

Take backups of the critical configuration files, such as `init.conf`, and `server1-obj.conf`, before making changes to these files. `Server1` is the name of the Application Server that you are about to configure.

1. Edit the `install_dir/domains/domain1/server1/config/init.conf` file and append the following lines:

On UNIX:

```
Init fn="load-modules"
shlib="appserver_install_dir/lib/webserver-plugin/platform/web_server/libpassthrough.so"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-passthrough" NativeThread="no"
Init fn="init-passthrough"
```

On Windows:

```
Init fn="load-modules"
shlib="c:/appserver_install_dir/lib/webserver-plugin/platform/web_server/libpassthrough.dll"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-passthrough" NativeThread="no"
Init fn="init-passthrough"
```

2. Edit the `install_dir/domains/domain1/server1/config/server1-obj.conf` and add the `AuthTrans` directive as shown here:

```
<Object name="default">
AuthTrans fn="match-browser" browser="*MSIE*" ssl-unclean-shutdown="true"
AuthTrans fn="auth-passthrough"
</Object>
```

3. Append the bin `PATH` of the Application Server with the path of the plug-in.
4. Restart the Sun Java System Application Server instance.

This completes the changes you need to make to Sun Java System Application Server. The procedures in the preceding section is to configure a single instance of the Application Server. To enable more instances to use the web server plug-in, you must make these changes in the configuration files of those Application Server instances.

---

#### NOTE

- For Solaris and Linux, the name of the plug-in library will be `libpassthrough.so`. For Windows, the name of the plug-in library will be `passthrough.dll`. However, for Microsoft IIS, the plug-in is named as `sun-passthrough.dll`.
  - The value of `keepAliveTimeout` should be the same in both the web server's `magnus.conf` file and the Application Server's `init.conf` file. If these values are different, the connection between the Application Server and web server may be closed prematurely, and 502 error messages will be printed in the `server.log` log file.
- 

## Configuring Application Server as a Proxy

Follow these instructions to configure Application Server to act as a proxy to another Application Server or web server. Make sure to take backups of critical configuration files, such as `magnus.conf`, `init.conf`, and `server1-obj.conf`, before making changes to these files.

1. Create a directory, `plugins/lbplugin/resource`, under `appserver_install_dir/domains/domain1/server1`.
2. Copy all `*.res` files from `appserver_install_dir/lib/webserver-plugin/platform/iws61` to `appserver_install_dir/domains/domain1/server1/plugins/lbplugin/resource`.
3. Edit the `magnus.conf` file, found under `appserver_install_dir/https-host.domain/config`, and append the following lines:

```
Init fn="load-modules"
shlib="appserver_install_dir/lib/webserver-plugin/platform/iws61/libpassthrough.so"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-passthrough" NativeThread="no"
Init fn="init-passthrough"
```

4. Edit the `server1-obj.conf` file, found under `appserver_install_dir/https-host.domain/config`, and add the `NameTrans` directive as shown:

```
<Object name="default">
NameTrans fn="assign-name" from="/*" name="passthrough"
...
</Object>
```

The `from="/*"` URI is the context root of a web application deployed on the remote servers, and `passthrough` corresponds to the name of the `<Object>` in `obj.conf`.

For example:

```
<Object name="default">
NameTrans fn="assign-name" from="( /webapp1|/webapp1/* )" name="passthrough"
...
</Object>
```

5. Add the following lines in the `obj.conf` file:

```
<Object name="passthrough">
ObjectType fn="force-type" type="magnus-internal/passthrough"
PathCheck fn="deny-existence" path="*/WEB-INF/*"
Service type="magnus-internal/passthrough" fn="service-passthrough"
servers="http://servername:port"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</Object>
```

*servername* is the hostname or IP address of the machine to which the proxy plug-in must route HTTP/S requests to, and *port* is the number of the port on which the server listens (this value is typically set to 80).

6. Include the path of the plug-in file in the `PATH` environment variable.
7. To enable these changes, use the following command to reconfigure the Application Server instance.

```
asadmin reconfig --user admin --password password --port 4848
--keepmanualchanges server1
```

8. Restart the Sun Java System Application Server instance.

## Configuring Microsoft IIS

Configuring the Microsoft Internet Information Services to use the web server plug-in involves configuring the web server plug-in for use with Microsoft IIS, and configuring Microsoft IIS to use the web server plug-in.

You can also configure server pools to handle multiple applications that run on different servers.

The following topics are covered in this section:

- [Configuring the Web Server Plug-in for IIS](#)
- [Configuring IIS to Use the Web Server Plug-in](#)
- [Configuring Multiple Server Pools](#)
- [Sample sun-passthrough.properties File](#)

## Configuring the Web Server Plug-in for IIS

To configure the web server plug-in for IIS, perform the following tasks:

1. Create a directory for the web server plug-in under the under the IIS `wwwroot` directory, by typing the following command, from the C:\ command line prompt:

```
md \Inetpub\wwwroot\sun-passthrough
```

2. Copy the plug-in files from the product CD to the `C:\Inetpub\wwwroot\sun-passthrough` directory.

By default, the plug-in files, after unzipping, will be located in:

```
appserver_install_dir\lib\webserver-plugin\windows\iis\
```

3. Create a directory, `lbplugin\resource`, under `web_server_install_dir\sun-passthrough`.
4. Copy all \*.res files from `appserver_install_dir\lib\webserver-plugin\windows\iis` to `web_server_install_dir\sun-passthrough\lbplugin\resource`.
5. Copy the `sun-passthrough.properties` file from `appserver_install_dir\lib\webserver-plugin\windows\iis` to `web_server_install_dir\sun-passthrough`.

6. Use a text editor to add the URL of the machine (on which Sun Java System Application Server is installed), to the

C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties file.

```
server=http://appservername:port
```

where, *appservername* is the hostname or IP address of machine on which Sun Java System Application Server is installed, and *port* is the number of the port on which it listens (this value is typically set to 80).

7. List the context roots you want Application Server to service in the C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties file.

These context roots should correspond to the context roots of applications deployed on Application Server. Requests to these context roots will be serviced by Application Server, while other requests are handled by the IIS web server. The command lines to pass requests to a web application is:

```
passthrough=/webapplication
```

where, */webapplication* is the context root of a web application. To pass all requests to Application Server, add the following line:

```
passthrough= /
```

You have now configured the web server plug-in in the Microsoft IIS root directory. To complete the process, you now need to configure Microsoft IIS to use the web server plug-in.

## Configuring IIS to Use the Web Server Plug-in

To configure IIS to use the web server plug-in, you need to open the Windows Internet Services Manager. The Internet Services Manager is located in the Administrative Tools folder in the Control Panel folder.

Open the Internet Services Manager, and perform the following tasks:

1. Select the web site for which you want to enable the plug-in. This web site is typically named the Default Web Site.
2. Right click on the web site and select Properties to open the Properties notebook.
3. Open the ISAPI Filters tab, click the Add button, and follow the steps given below, to add a new ISAPI filter:
  - a. In the Filter Name field, enter Application Server

- b. In the Executable field, type  
`C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll`
        - c. Click OK, and close the Properties notebook.
  4. You now need to create and configure a new virtual directory. Follow the steps given below to create and configure a new virtual directory:
    - a. Right click on the default web site, select New, and then Virtual Directory. The Virtual Directory Creation Wizard opens.
    - b. In the Alias field, type `sun-passthrough`.
    - c. In the Directory field, type `C:\Inetpub\wwwroot\sun-passthrough`
    - d. Ensure that you check the Execute Permission checkbox and that all other permission checkboxes are left unchecked.
    - e. Click Finish.
  5. Add the path of `sun-passthrough.dll` file to the PATH environment variable of the bin path of Application Server.
  6. You need to stop and start the web server, for your new settings to take effect. To stop the web server, right click on the web site and select Stop. To start the web server, right click on the web site and select Start.

Next, type the following in a web browser, to access the web application context root:

`http://webservername/webapplication`

where, *webservername* is the hostname or IP address of the web server and */webapplication* is the context root that you listed in the

`C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` file to verify that the web server, web server plug-in, and Application Server are operating correctly.

## Configuring Multiple Server Pools

It is possible to partition your web applications across multiple Application Servers (that is, you run some applications on one set of servers and other applications on another set of servers) by configuring server pools in the `sun-passthrough.properties` file. For each server pool, choose a unique name, comprised of letters and numbers. Once you complete the steps for installing and configuring the web server plug-in for Microsoft IIS, as given in the section

**“Configuring Microsoft IIS” on page 324, edit the**

`C:\Inetpub\wwwroot\plugins\lbplugins\sun-passthrough.properties` file and prefix the relevant server and passthrough property lines with the unique name that you choose for the server pool. Place a period (.) after the server pool name.

For example, the following lines from the `sun-passthrough.properties` file define two server pools. The first server pool consists of `server-a` and the services requests for context root `/app1`. The second server pool consists of `server-b` and the services requests for `/app2` and `/app3` context roots.

```
server=http://server-a
passthrough=/app1
serverpool2.server=http://server-b
serverpool2.passthrough=/app2
serverpool2.passthrough=/app
```

## Sample sun-passthrough.properties File

```
# Sun Java System Application Server web server plugin for IIS
#
# This file is used to configure the Sun Java System Application Server web
server
# plugin for IIS.  Lines beginning with a '#' are ignored.
# server
#
# The server property specifies the URL of an application server.  If
multiple
# server properties are given, the plugin will distribute load across the
# specified application servers.
#
server=http://localhost:8080
# passthrough
#
# The passthrough property specifies the context root (virtual directory)
of a
```

```

# web application.  Requests for the given context root will be passed to
the
# application server for processing.  If 'passthrough=/' is specified, all
# requests will be passed to the application server for processing.
#
# passthrough properties should be ordered from most to least specific.  For
# example, 'passthrough=/apps/app1' should appear before
# 'passthrough=/apps'.
#
# Multiple passthrough properties are allowed.
#
#passthrough=/webapp
#passthrough=/servlets
#passthrough=*.jsp
passthrough=/
# prefix
#
# The prefix property specifies the IIS virtual directory that contains the
# plugin DLL, sun-passthrough.dll.
#
prefix=/sun-passthrough
# error-url
#
# The error-url property specifies the URL of a page to redirect the client
to
# when the application server is unavailable.
#
#error-url=/badgateway.htm
# It is possible to configure multiple server pools by prefixing the server
# and passthrough property names with a pool name followed by a period
# ('.').

```



```
# Pool names can be any sequence of letters and numbers.
#
# For example, the following properties define two server pools. One server
# pool will service the web applications at '/app1' and the other will
# service
# the web applications at '/app2' and '/app3':
#
#serverpool1.server=http://server-a
#serverpool1.passthrough=/app1
#
#serverpool2.server=http://server-b
#serverpool2.passthrough=/app2
#serverpool2.passthrough=/app3
```

## Configuring Apache Web Server

This section describes how you can compile the Apache source code and configure your installation of Apache Web Server to work with Sun Java System Application Server.

To enable HTTP requests to be sent to Sun Java System Application Server, you must compile the Apache sources to use the `mod_proxy` module, and then modify the `httpd.conf` file.

This section contains information on the following procedures:

- [Minimum Requirements](#)
- [Compiling Apache With `mod\_proxy` Module](#)
- [Modifying the `httpd.conf` File](#)
- [Starting And Stopping Apache](#)

## Minimum Requirements

You must meet the following requirements to successfully compile Apache web server and use the `mod_proxy` plug-in.

- `apache_1.3.27` (sources)
- Sun's C Compiler for Solaris 8 and 9.
- Sun Java System Application Server, Standard Edition installables.
- On Solaris 8 and 9, ensure that `CC` and `make` are in the `PATH`.

## Compiling Apache With `mod_proxy` Module

1. Download the latest Apache source distribution with built-in `mod_proxy` module from `www.apache.org`

Unpack the source distribution. The source distribution comes as a compressed archive. If you are installing Apache 1.3.27, the source distribution archive will read `apache_1.3.27.tar.gz`.

2. De-compress and untar the archive using the following command:

```
$ tar -zxvf apache_1.3.27.tar.gz
```

This command will create a directory called `apache_1.3.27` in your current working directory.

3. You now need to configure your environment to compile the Apache source code. The source distribution comes with a script called `configure`, which checks your environment for the necessary support files (such as headers, shared libraries and utility programs) that are required to successfully compile Apache.

To configure you environment, go to the Apache source directory and proceed with the following steps:

- a. Make sure that the following paths exist, while installing Apache on Solaris:

- `CC=/opt/SUNWspro/bin/cc`

Where `/opt/SUNWspro/bin` is the path where `cc` is installed. Make sure that this is in your `PATH`.

- Make sure that `/usr/ccs/bin` is in your `PATH`.

**b. Run the following command:**

```
./configure --enable-module=proxy --prefix=/usr/local/apache
```

The path specified in the `prefix` argument indicates where you wish to install Apache. This is a variable and you can specify the path where you want to install Apache.

This command will output several lines on the screen. Essentially this command creates the Makefiles for the build according to your system configuration. If there are errors in `configure`, you may be missing some header files or utility programs which you must install before proceeding.

4. After the `configure` script runs successfully, you can compile Apache using the `make` command, as follows:

```
make
```

This command will output several lines on the screen indicating that the process is compiling Apache source code and linking Apache. This process should normally conclude without errors. However if any errors occur, please check if all the library files and utility programs of Apache have been properly downloaded.

5. You now need to install Apache. Apache installs itself in the `/usr/local/apache` directory (or any other directory that you specify). To install Apache, run the following command:

```
make install
```

If this command executes successfully, your system now has Apache installed. You should see Apache's installation files in the following directory:

```
/usr/local/apache
```

The main configuration file, called `httpd.conf`, will be installed in the `/usr/local/apache/conf` directory.

---

<b>NOTE</b>	<code>mod_proxy</code> lacks the SSL transport and load balancing features of Application Server's Reverse Proxy Plug-in. Also, it is not possible for web applications to obtain the client's IP address or SSL client certificate when using <code>mod_proxy</code> .
-------------	---

---

## Modifying the httpd.conf File

Apache is configured through the file, `httpd.conf`. This file consists of a number of Apache directives, which determine the various operating parameters of the Apache server. For a simple installation of Apache, you will need to modify the following few directives:

```
ServerRoot "/usr/local/apache"
Port 5000
```

`ServerRoot` is the path in which you installed Apache.

Apache is now configured for default behavior and web serving. Next, you must add the following Application Server specific directives to the `httpd.conf` file to enable Apache to forward HTTP requests to Sun Java System Application Server:

```
<IfModule mod_proxy.c>
ProxyPass /application http://<appserver.some.domain>:<port>/
ProxyPassReverse /application http://<appserver.some.domain>:<port>/
</IfModule>
```

Here, `<appserver.some.domain>:<port>` should be replaced by the URL address of your Sun Java System Application Server. Duplicate these two lines for each web application context root, where `/application` is the web application context root and `http://<appserver.some.domain>:<port>` is the URL of the Sun Java System Application Server.

This completes your configuration of Apache web server.

## Starting And Stopping Apache

Apache comes bundled with a script titled `apachectl` that facilitates starting, stopping and restarting Apache. Run the follow command to start Apache

```
$ /usr/local/apache/bin/apachectl start
```

To stop apache, run the following command:

```
use /usr/local/apache/bin/apachectl stop
```

After you start, you can test your installation of Apache. Once Apache is running, type the following address in your web browser: `http://localhost/`. If your installation was successful and Apache is running, you should see a test page displaying a message to the effect.

# Deploying Applications

This chapter describes how to deploy various Sun Java System Application Server modules and applications.

Sun Java System Application Server modules and applications include J2EE standard elements and Sun Java System Application Server specific elements. Only Sun Java System Application Server specific elements are described in detail in this chapter.

To know about packaging and assembling modules and applications for deployment, see the *Sun Java System Application Server Developer's Guide*.

This chapter includes the following topics:

- [About J2EE Modules](#)
- [About J2EE Applications](#)
- [J2EE Standard Descriptors](#)
- [Application Server Descriptors](#)
- [Naming Standards](#)
- [Deployment Directory Structure](#)
- [Runtime Environments](#)
- [About Classloaders](#)
- [Deploying Modules and Applications](#)
- [The Application Deployment Descriptor Files](#)

# About J2EE Modules

A J2EE module is a collection of one or more J2EE components of the same container type with deployment descriptors of that type. One descriptor is J2EE standard, the other is Sun Java System Application Server specific. Types of J2EE modules are as follows:

- **Web Application Archive (WAR):** A web application is a collection of servlets, HTML pages, classes, and other resources that can be bundled and deployed to several J2EE application servers. A WAR file can consist of the following items: servlets, JSPs, JSP tag libraries, utility classes, static pages, client-side applets, beans, bean classes, and deployment descriptors (`web.xml` and optionally `sun-web.xml`).
- **EJB JAR File:** The EJB JAR file is the standard format for assembling enterprise beans. This file contains the bean classes (home, remote, local, and implementation), all of the utility classes, and the deployment descriptors (`ejb-jar.xml` and optionally `sun-ejb-jar.xml`). If the EJB is an entity bean with container managed persistence, a CMP deployment descriptor, `sun-cmp-mapping.xml`, may be included as well.
- **Application (RMI/IIOP) Client JAR File:** An RMI/IIOP Client is a Sun Java System Application Server specific type of J2EE client. An RMI/IIOP Client supports the standard J2EE Application Client specifications, and in addition, supports direct access to the Sun Java System Application Server. Its deployment descriptors are `application-client.xml` and optionally `sun-application-client.xml`.
- **Resource RAR File:** RAR files apply to J2EE CA connectors. A connector module is like a device driver. It is a portable way of allowing EJBs to access a foreign enterprise system. Each Sun Java System Application Server connector has a J2EE XML file, `ra.xml`. A connector must also have a Sun Java System Application Server deployment descriptor, `sun-ra.xml`.

Package definitions must be used in the source code of all modules so the classloader can properly locate the classes after the modules have been deployed.

Because the information in a deployment descriptor is declarative, it can be changed without requiring modifications to source code. At run time, the J2EE server reads this information and acts accordingly.

EJB JAR and Web modules can also be assembled as separate `.jar` or `.war` files and deployed separately, outside of any application.

# About J2EE Applications

A J2EE application is a logical collection of one or more J2EE modules tied together by application deployment descriptors. Components can be assembled at either the module or the application level. Components can also be deployed at either the module or the application level.

Components are assembled into modules and then assembled into a Sun Java System Application Server application `.ear` file ready for deployment.

Each module has a Sun Java System Application Server deployment descriptor and a J2EE deployment descriptor. The Sun Java System Application Server Administration interface uses the deployment descriptors to deploy the application components and to register the resources with the Sun Java System Application Server.

An application consists of one or more modules, an optional Sun Java System Application Server deployment descriptor, and a required J2EE application deployment descriptor. All items are assembled, using the Java ARchive (`.jar`) file format, into one file with an extension of `.ear`.

## J2EE Standard Descriptors

The J2EE platform provides assembly and deployment facilities. These facilities use JAR files as the standard package for components and applications, and XML-based deployment descriptors for customizing parameters. For more information on the J2EE assembly and deployment process, see *Developing Enterprise Applications with the J2EE*, v 1.0, Chapter 7.

The J2EE standard deployment descriptors are described in the J2EE specification, v1.3.

To check the correctness of these deployment descriptors prior to deployment, see the information on the deployment descriptor verifier in the *Sun Java System Application Server Developer's Guide*.

[Table 13-1 on page 336](#) shows where to find more information about J2EE standard deployment descriptors. The left column lists the deployment descriptors, and the right column lists where to find more information about those descriptors.

**Table 13-1** J2EE Standard Descriptors

Deployment Descriptor	Where to Find More Information
application.xml	Java 2 Platform Enterprise Edition Specification, v1.3, Chapter 8, "Application Assembly and Deployment - J2EE:application XML DTD"
web.xml	Java Servlet Specification, v2.3 Chapter 13, "Deployment Descriptor," and JavaServer Pages Specification, v1.2, Chapter 7, "JSP Pages as XML Documents," and Chapter 5, "Tag Extensions"
ejb-jar.xml	Enterprise JavaBeans Specification, v2.0, Chapter 16, "Deployment Descriptor"
application-client.xml	Java 2 Platform Enterprise Edition Specification, v1.3, Chapter 9, "Application Clients - J2EE:application-client XML DTD"
ra.xml	Java 2 Enterprise Edition, J2EE Connector Architecture Specification, v1.0, Chapter 10, "Packaging and Deployment."

You can find specifications here:

<http://java.sun.com/products/>

# Application Server Descriptors

Sun Java System Application Server uses additional deployment descriptors for configuring features specific to the Sun Java System Application Server. These are optional except for the `sun-ra.xml` file, which is required for a connector module.

To check the correctness of these deployment descriptors prior to deployment, see the information on the deployment descriptor verifier in the *Sun Java System Application Server Developer's Guide*.

[Table 13-2 on page 337](#) shows where to find more information about Sun Java System Application Server deployment descriptors. The left column lists the deployment descriptors, and the right column lists where to find more information about those descriptors.



**Table 13-2** Sun Java System Application Server Descriptors

Deployment Descriptor	Where to Find More Information
sun-application.xml	<a href="#">“The Application Deployment Descriptor Files” on page 352.</a>
sun-web.xml	<i>Sun Java System Application Server Developer's Guide to Web Applications</i>
sun-ejb-jar.xml and sun-cmp-mapping.xml	<i>Sun Java System Application Server Developer's Guide to Enterprise Java Beans</i>
sun-application-client.xml and sun-acc.xml	<i>Sun Java System Application Server Developer's Guide to Clients</i>
sun-ra.xml	<i>Sun Java System J2EE CA Service Provider Implementation Administrator's Guide</i>

---

**NOTE** The Sun Java System Application Server deployment descriptors must have 600 level access privileges on UNIX systems.

---

The DTD schema files for all the Sun Java System Application Server deployment descriptors are located in the *install\_dir/appserv/lib/dtds* directory.

## Naming Standards

Names of applications and individually deployed EJB JAR, WAR, and connector RAR modules (as specified by the `name` attributes in the `server.xml` file) must be unique in the Sun Java System Application Server. If you do not explicitly specify a name, the default name is the first portion of the file name (without the `.war` or `.jar` extension). For details about `server.xml`, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

Modules of different types can have the same name within an application, because when the application is deployed, the directories holding the individual modules are named with `_jar`, `_war` and `_rar` suffixes. Modules of the same type within an application must have unique names. In addition, database schema file names must be unique within an application.

Using a Java package-like naming scheme is recommended for module filenames, EAR filenames, module names as found in the `<module-name>` portion of the `ejb-jar.xml` files, and EJB names as found in the `<ejb-name>` portion of the `ejb-jar.xml` files. The use of this package-like naming scheme ensures that name collisions do not occur. The benefits of this naming practice apply not only to the Sun Java System Application Server, but to other J2EE application servers as well.

JNDI lookup names for EJBs must also be unique. Here too, establishing a consistent naming convention may help. For example, appending the application name and the module name to the EJB name would be one way to guarantee unique names. In this case, `mycompany.pkging.pkgingEJB.MyEJB` would be the JNDI name for an EJB in the module `pkgingEJB.jar`, which is packaged in the application `pkging.ear`.

Make sure your package and file names do not contain spaces or characters that are illegal for your operating system.

## Deployment Directory Structure

When you deploy an application, the directories holding the individual modules are named with `_jar`, `_war` and `_rar` suffixes. If you use the `asadmin deploydir` command to deploy a directory instead of an EAR file, your directory structure must follow this convention.

Module and application directory structures follow the structure outlined in the J2EE specification.

Here is an example directory structure of a simple application containing a web module, an EJB module, and a client module.

```

+ converter_1/
|--- converterClient.jar
|---+ META-INF/
|   |--- MANIFEST.MF
|   |--- application.xml
|   '--- sun-application.xml
|---+ war-ic_war/
|   |--- index.jsp
|   |---+ META-INF/
|   |   |--- MANIFEST.MF
|   |   '---+ WEB-INF/
|   |       |--- web.xml
|   |       '--- sun-web.xml
|---+ ejb-jar-ic_jar/
|   |--- Converter.class
|   |--- ConverterBean.class
|   |--- ConverterHome.class
|   |---+ META-INF/
|   |   |--- MANIFEST.MF
|   |   |--- ejb-jar.xml
|   |   '--- sun-ejb-jar.xml
|---+ app-client-ic_jar/
|   |--- ConverterClient.class
|   |---+ META-INF/
|   |   |--- MANIFEST.MF
|   |   |--- application-client.xml
|   |   '--- sun-application-client.xml

```

Here is an example directory structure of an individually deployed connector module.

```

+ MyConnector/
|--- readme.html
|--- ra.jar
|--- client.jar
|--- win.dll
|--- solaris.so
|---+ META-INF/
|   |--- MANIFEST.MF
|   |--- ra.xml
|   '--- sun-ra.xml

```

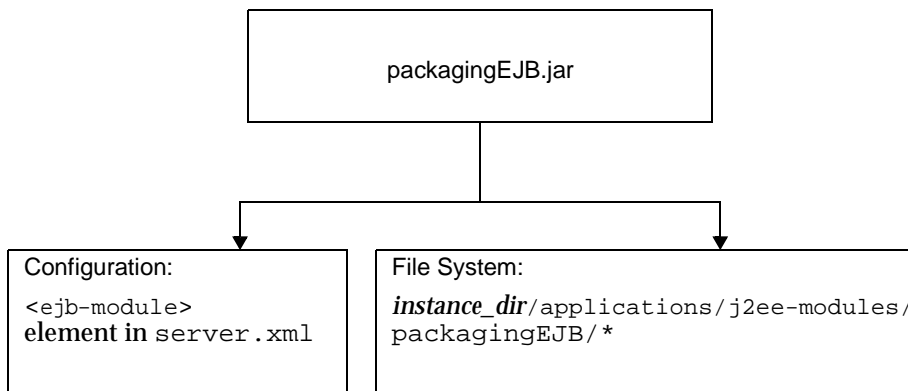
# Runtime Environments

Whether you deploy a component as an individually deployed module or as an application, deployment affects both the file system and the server configuration. See the “[Module Runtime Environment](#)” and “[Application Runtime Environment](#)” figures.

## Module Runtime Environment

The following figure, “[Module Runtime Environment](#),” illustrates the environment for individually deployed module-based deployment.

**Figure 13-1** Module Runtime Environment



For file system entries, modules are extracted as follows:

```

instance_dir/applications/j2ee-modules/module_name
instance_dir/generated/ejb/j2ee-modules/module_name
instance_dir/generated/jsp/j2ee-modules/module_name
  
```

The `generated/ejb` directory contains stubs and ties; the `generated/jsp` directory contains compiled JSPs.

Lifecycle modules are extracted as follows:

```

instance_dir/applications/lifecycle-modules/module_name
  
```

Configuration entries are added in `server.xml` as follows:

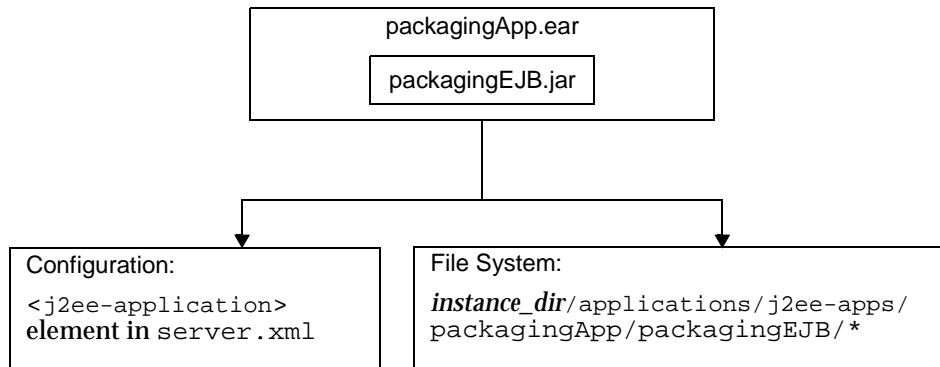
```
<server>
  <applications>
    <type-module>
      ...module configuration...
    </type-module>
  </applications>
</server>
```

The *type* of the module in `server.xml` can be `lifecycle`, `ejb`, `web`, or `connector`. For details about `server.xml`, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

## Application Runtime Environment

Figure 13-2 illustrates the environment for application-based deployment.

**Figure 13-2** Application Runtime Environment



For file system entries, applications are extracted as follows:

```
instance_dir/applications/j2ee-apps/app_name
instance_dir/generated/ejb/j2ee-apps/app_name
instance_dir/generated/jsp/j2ee-apps/app_name
```

The `generated/ejb` directory contains stubs and ties; the `generated/jsp` directory contains compiled JSPs.

Configuration entries are added in `server.xml` as follows:

```
<server>
  <applications>
    <j2ee-application>
      ...application configuration...
    </j2ee-application>
  </applications>
</server>
```

For details about `server.xml`, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

## About Classloaders

Understanding Sun Java System Application Server classloaders can help you determine where and how you can position supporting JAR and resource files for your modules and applications.

In a Java Virtual Machine (JVM), the classloaders dynamically load a specific java class file needed for resolving a dependency. For example, when an instance of `java.util.Enumeration` needs to be created, one of the classloaders loads the relevant class into the environment. For a more detailed discussion on Classloaders, see the *Sun Java System Application Server Developer's Guide*.

## Deploying Modules and Applications

This section describes the different ways to deploy J2EE applications and modules to the Sun Java System Application Server. It covers the following topics:

- [Deployment Names and Errors](#)
- [The Deployment Life Cycle](#)
- [Using Availability Features \(Enterprise Edition\)](#)
- [Deploying a Module or Application](#)
- [Deploying a WAR Module](#)
- [Deploying an EJB JAR Module](#)
- [Deploying a Lifecycle Module](#)
- [Deploying an RMI/IIOP Client](#)

- [Deploying a J2EE CA Resource Adapter](#)
- [Deploying Static Content](#)
- [Access to Shared Frameworks](#)

## Deployment Names and Errors

A unique name is generated in the `server.xml` file when you deploy an application or module. Do not change this name. During deployment, the server detects any name collisions and does not load an application or module having a non-unique name. Messages are sent to the server log when this happens. For more about naming, see [“Naming Standards” on page 337](#).

If an error occurs during deployment, the application or module is not deployed. If a module within an application contains an error, the entire application is not deployed.

For details about `server.xml`, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

## The Deployment Life Cycle

After an application is initially deployed, it may be modified and reloaded, redeployed, disabled, reenabled, and finally undeployed (removed from the server). This section covers the following topics related to the deployment life cycle:

- [Dynamic Deployment](#)
- [Disabling a Deployed Application or Module](#)
- [Dynamic Reloading](#)

### Dynamic Deployment

You can deploy, redeploy, and undeploy an application or module without restarting the server. This is called dynamic deployment. Note that if you change the deployment settings when you redeploy you have to restart the server.

Although primarily for developers, dynamic deployment can be used in operational environments to bring new applications and modules online without requiring a server restart. Whenever a redeployment is done, the sessions at that transit time become invalid. The client must restart the session.

## Disabling a Deployed Application or Module

You can disable a deployed application or module without removing it from the server. Each application and module has an `enabled` attribute in the `server.xml` file and a corresponding option in the Administration interface, which you can change. For details about `server.xml`, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

## Dynamic Reloading

If dynamic reloading is enabled, you do not have to redeploy an application or module when you change its code. All you have to do is copy the changed class files into the deployment directory for the application or module. The server checks for changes periodically and redeploys the application, automatically and dynamically, with the changes.

This is useful in a development environment, because it allows code changes to be tested quickly. Dynamic reloading is not recommended for a production environment, however, because it may degrade performance. In addition, whenever a reload is done, the sessions at that transit time become invalid. The client must restart the session.

To enable dynamic reloading, you can do one of the following:

### *Use the Administration interface*

1. Open the Applications component under your server instance.
2. Go to the Applications page.
3. Check the Reload Enabled box to enable dynamic reloading.
4. Enter a number of seconds in the Reload Poll Interval field to set the interval at which applications and modules are checked for code changes and dynamically reloaded.
5. Click on the Save button.
6. Go to the server instance page and select the Apply Changes button.

### *Edit the server.xml file*

Edit the following attributes of the `server.xml` file's `applications` element:

- `dynamic-reload-enabled="true"` enables dynamic reloading.
- `dynamic-reload-poll-interval-in-seconds` sets the interval at which applications and modules are checked for code changes and dynamically reloaded.



For details about `server.xml`, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

In addition, to load new servlet files, reload EJB related changes, or reload deployment descriptor changes, you must do the following:

1. Create an empty file named `.reload` at the root of the deployed application:

```
instance_dir/applications/j2ee-apps/app_name/.reload
```

or individually deployed module:

```
instance_dir/applications/j2ee-modules/module_name/.reload
```

2. Explicitly update the `.reload` file's timestamp (`touch .reload` in UNIX) each time you make the above changes.

For JSPs, changes are reloaded automatically at a frequency set in the `reload-interval` property of the `jsp-config` element in the `sun-web.xml` file. To disable dynamic reloading of JSPs, set `reload-interval="-1"`.

## Using Availability Features (Enterprise Edition)

When you deploy an application, you can choose to enable availability for the application (availability enabled set to true) or you can choose Specified by Container to automatically use the availability setting specified in the web or EJB container level. If availability is enabled, the application can fail over HTTP sessions and stateful session beans. If you want availability enabled for an application, it must also be enabled at all higher levels (server instance and web container or EJB container) as well.

For more information about availability, see [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\).”](#)

## Tools for Deployment

This section discusses the various tools that can be used to deploy modules and applications. The deployment tools include:

- [The asadmin Utility](#)
- [The Administration Interface](#)
- [Sun ONE Studio](#)

## The asadmin Utility

You can use the `asadmin` utility to deploy or undeploy applications and individually deployed modules on local servers. Concurrent deployment on multiple machines is not supported. This section describes the `asadmin` utility only briefly.

To deploy a lifecycle module, see [“Deploying a Lifecycle Module” on page 349](#).

### *asadmin deploy*

The `asadmin deploy` command deploys a WAR, JAR, RAR, or EAR file. To deploy an application, specify `--type application` in the command. To deploy an individual module, specify `--type ejb, web, connector`. The syntax is as follows, with defaults shown for optional parameters that have them:

```
asadmin deploy --user admin_user [--password admin_password] [--host localhost]
[--port 4848] [--secure | -s] [--passwordfile file_name] [--virtualservers
virtual_servers] [--type application|ejb|web|connector] [--contextroot contextroot]
[--force=true] [--precompilejsp=false] [--verify=false] [--name
component_name] [--upload=true] [--availabilityenabled] [--retrieve local_dirpath]
[--instance instance_name] filepath
```

---

**NOTE** The `availabilityenabled` option is only available in the Sun Java System Application Server, Enterprise Edition.

---

For example, the following command deploys an individual EJB module:

```
asadmin deploy --user jadams --password secret --host localhost --port 4848
--type ejb --instance server1 packagingEJB.jar
```

### *asadmin deploydir*

The `asadmin deploydir` command deploys an application or module in an open directory structure. The structure must be as specified in [“Deployment Directory Structure” on page 338](#). The location of the *dirpath* under *instance\_dir/applications/j2ee-apps* or *instance\_dir/applications/j2ee-modules* determines whether it is an application or individually deployed module. The syntax is as follows, with defaults shown for optional parameters that have them:

```
asadmin deploydir --user admin_user [--password admin_password] [--host
localhost] [--port 4848] [--secure | -s] [--passwordfile file_name]
[--virtualservers virtual_servers] [--type application|ejb|web|connector]
[--contextroot contextroot] [--force=true] [--precompilejsp=false]
[--verify=false] [--availabilityenabled] [--name component_name] [--instance
instance_name] dirpath
```

---

**NOTE** The `availabilityenabled` option is only available in the Sun Java System Application Server, Enterprise Edition.

---

For example, the following command deploys an individual EJB module:

```
asadmin deploydir --user jadams --password secret --host localhost --port
4848 --type ejb --instance server1 packagingEJB
```

### *asadmin undeploy*

The `asadmin undeploy` command undeploys an application or module. To undeploy an application, specify `--type app` in the command. To undeploy a module, specify `--type ejb`, `web`, or `connector`. The syntax is as follows, with defaults shown for optional parameters that have them:

```
asadmin undeploy --user admin_user [--password admin_password] [--host
localhost] [--port 4848] [--secure | -s] [--passwordfile file_name] [--type
application|ejb|web|connector] [--instance instance_name] component_name
```

For example, the following command undeploys an individual EJB module:

```
asadmin undeploy --user jadams --password secret --host localhost --port
4848 --type ejb --instance server1 packagingEJB
```

## The Administration Interface

You can use the Administration interface to deploy modules and applications to both local and remote Sun Java System Application Server sites. To use this tool, follow these steps:

1. Open the Applications component under your server instance.
2. Go to the Enterprise Applications, Web Applications, Connector Modules, or EJB Modules page.
3. Click on the Deploy button.
4. Enter the full path to the module or application (or click on Browse to find it), then click on the OK button.
5. Enter the module or application name.

You can also redeploy the module or application if it already exists by checking the appropriate box. This is optional.

6. Assign the application or module to one or more virtual servers by checking the boxes next to the virtual server names.

7. Choose whether to enable availability, which enables failover at this level if it is enabled at higher levels (Enterprise Edition only).
8. Click on the OK button.

To deploy a lifecycle module, see [“Deploying a Lifecycle Module” on page 349](#).

## Sun ONE Studio

You can use Sun ONE Studio to deploy J2EE applications and modules. For more information about using Sun ONE Studio, see the *Sun ONE Studio, Enterprise Edition Tutorial*.

---

<b>NOTE</b>	In Sun ONE Studio, deploying a module or application is referred to as <i>executing</i> it. Execution also includes making sure the server is running and displaying the correct URL to activate the module or application.
-------------	---

---

## Deploying a Module or Application

You can deploy applications or individual modules that are independent of applications. The runtime and file system implications of application-based or individual module-based deployment are described in [“Runtime Environments” on page 340](#).

Individual module-based deployment is preferable when components need to be accessed by:

- Other modules
- J2EE Applications
- RMI/IIOP clients (Module-based deployment allows shared access to a bean from an RMI/IIOP client, a servlet, or an EJB.)

Modules can be combined into an EAR file and then deployed as a single module. This is similar to deploying the modules of the EAR independently.

## Deploying a WAR Module

You deploy a WAR module in one of the ways described in [“Tools for Deployment” on page 345](#).

You can keep the generated source for JSPs by adding the `-keepgenerated` property to the `jsp-config` element in `sun-web.xml`. If you include this property when you deploy the WAR module, the generated source is kept in `instance_dir/generated/jsp/j2ee-apps/app_name/module_name` if it is in an application or `instance_dir/generated/jsp/j2ee-modules/module_name` if it is in an individually deployed web module. For more information about the `-keepgenerated` property, see the *Sun Java System Application Server Developer's Guide to Web Applications*.

## Deploying an EJB JAR Module

You deploy an EJB JAR module in one of the ways described in “Tools for Deployment” on page 345.

You can keep the generated source for stubs and ties by adding the `-keepgenerated` flag to the `rmic-options` attribute of the `java-config` element in `server.xml`. If you include this flag when you deploy the EJB JAR module, the generated source is kept in `instance_dir/generated/ejb/j2ee-apps/app_name/module_name` if it is in an application or `instance_dir/generated/ejb/j2ee-modules/module_name` if it is in an individually deployed EJB JAR module. For more information about the `-keepgenerated` flag, see the *Sun Java System Application Server Administrator's Configuration File Reference*. For more information about `rmic`, see the Java 2 SDK Tools and Utilities documentation at <http://java.sun.com/docs/index.html>.

## Deploying a Lifecycle Module

For general information about lifecycle modules, see the *Sun Java System Application Server Developer's Guide*.

You can deploy a lifecycle module using the following tools:

- [The asadmin Utility](#)
- [The Administration Interface](#)

### The asadmin Utility

To deploy a lifecycle module, use the `asadmin create-lifecycle-module` command. The syntax is as follows, with defaults shown for optional parameters that have them:

```
asadmin create-lifecycle-module --user admin_user [--password admin_password]
[--host localhost] [--port 4848] [--secure | -s] [--passwordfile file_name]
[--instance instance_name] --classname classname [--classpath classpath]
[--loadorder load_order_number] [--failurefatal=false] [--enabled=true]
[--description text_description] [--property (name=value)[:name=value]*] modulename
```

For example:

```
asadmin create-lifecycle-module --user jadams --password secret --host
localhost --port 4848 --instance server1 --classname RMIServer MyRMIServer
```

To undeploy a lifecycle module, use the `asadmin delete-lifecycle-module` command. The syntax is as follows, with defaults shown for optional parameters that have them:

```
asadmin delete-lifecycle-module --user admin_user [--password admin_password]
[--host localhost] [--port 4848] [--secure | -s] [--passwordfile file_name]
[--instance instance_name] module_name
```

For example:

```
asadmin delete-lifecycle-module --user jadams --password secret --host
localhost --port 4848 --instance server1 MyRMIServer
```

To list the lifecycle modules that are deployed on a server instance, use the `asadmin list-lifecycle-modules` command. The syntax is as follows, with defaults shown for optional parameters that have them:

```
asadmin list-lifecycle-modules --user admin_user [--password admin_password]
[--host localhost] [--port 4848] [--secure | -s] [--passwordfile file_name]
instance_name
```

For example:

```
asadmin list-lifecycle-module --user jadams --password secret --host
localhost --port 4848 server1
```

## The Administration Interface

You can also use the Administration interface to deploy a lifecycle module. Follow these steps:

1. Open the Applications component under your server instance.
2. Go to the Life Cycle Modules page.
3. Click on the Deploy button.

4. Enter the following information:
  - Name (required) - The name of the life cycle module.
  - Class Name (required) - The fully qualified name of the life cycle module's class file.
  - Classpath (optional) - The classpath for the life cycle module. Specifies where the module is located. The default location is under the application root directory.
  - Load Order (optional) - Determines the order in which life cycle modules are loaded at startup. Modules with smaller integer values are loaded sooner. Values can range from 101 to the operating system's `MAXINT`. Values from 1 to 100 are reserved.
  - Failure Fatal (optional) - Determines whether the server is shut down if the life cycle module fails. The default is `false`.
  - Enable (optional) - Determines whether the life cycle module is enabled. The default is `true`.
5. Click OK.

## Deploying an RMI/IIOP Client

Deployment is only necessary for clients that communicate with EJBs. Deploying an RMI/IIOP client is a three-step process:

1. Deploy the EAR or EJB JAR to be accessed by the RMI/IIOP client.
2. Assemble the necessary client files and deploy the client.
3. After deployment, a client JAR file is created in the following location for an application:

`instance_dir/applications/j2ee-apps/app_name/app_nameClient.jar`

or in the following location for an individually deployed module:

`instance_dir/applications/j2ee-modules/module_name/module_nameClient.jar`

The client JAR contains the ties and necessary classes for the RMI/IIOP client. Copy this file to the client machine, and set the `APPCPATH` environment variable on the client to point to this JAR.

You are now ready to run the client. For more information, see the *Sun Java System Application Server Developer's Guide to Clients*.

## Deploying a J2EE CA Resource Adapter

You deploy a connector module in one of the ways described in [“Tools for Deployment” on page 345](#).

## Deploying Static Content

Static content (HTML, images, etc.) can be hosted both on the web server and on the Sun Java System Application Server. However, when a WAR is registered, the static content gets deployed on the application server. All of the samples shipped with Sun Java System Application Server host the static content on the application server.

For example, to access a static file `index.html` on the application server, use:  
`http://server:port/tcontext_root/index.html`

## Access to Shared Frameworks

When J2EE applications and modules use shared framework classes (such as components and libraries) the classes can be put in the path for the System Classloader or the Common Classloader rather than in an application or module. If you assemble a large, shared library into every module that uses it, the result is a huge file that takes too long to register with the server. In addition, several versions of the same class could exist in different classloaders, which is a waste of resources.

For more information about the system classloader, see [“About Classloaders” on page 342](#).

## The Application Deployment Descriptor Files

Sun Java System Application Server applications include two deployment descriptor files:

- A J2EE standard file (`application.xml`), described in the Java Servlet Specification, v2.3, Chapter 13, “Deployment Descriptors.”
- An optional Sun Java System Application Server specific file (`sun-application.xml`), described in this section.

For more information on application deployment descriptor files, see the *Sun Java System Application Server Developer's Guide*.



# Managing HTTP Server Features and Virtual Servers

Chapter 14, “Configuring HTTP Features”

Chapter 15, “Using Virtual Servers”

Chapter 16, “Managing Virtual Server Content”



# Configuring HTTP Features

This chapter describes how to configure preferences for your HTTP-related features of your Sun Java System Application Server. For preferences related to virtual servers and HTTP listeners, see [Chapter 15, “Using Virtual Servers.”](#)

This chapter includes the following topics:

- [About the HTTP Features](#)
- [Configuring the File Cache](#)
- [Tuning Your Server for Performance](#)
- [Configuring HTTP Quality of Service](#)
- [Adding and Using Thread Pools](#)
- [Editing Advanced Settings](#)
- [Configuring MIME Types](#)

## About the HTTP Features

The Sun Java System Application Server HTTP features include setting performance levels for application server instances, setting performance tuning-related parameters, and using the file cache to improve performance. These settings are stored in two configuration files: `init.conf` and `server.xml`. You edit the `init.conf` settings on the Advanced Settings page. For more information, see [“Editing Advanced Settings” on page 359.](#)

Other properties you edit are stored in the `server.xml` file, in the `http-service` element. For more information on both the `init.conf` file and the `server.xml` file, see the *Sun Java System Application Server Administrator’s Configuration File Reference*.

## Configuring the File Cache

The Sun Java System Application Server uses a file cache to serve static information faster. The file cache contains information about files and static file content. The file cache also caches information that is used to speed up processing of server-parsed HTML.

The file cache is turned on by default. The file cache settings are contained in a file called `nsfc.conf`. This file is present only if file cache parameters have been changed from their defaults. For more information on `nsfc.conf`, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

To configure the file cache:

1. In the left pane, click HTTP Server
2. Click the File Caching tab
3. Enter the desired value in the fields.
4. Click OK.

For more information on using the file cache to improve performance, see the *Sun Java System Application Server Performance Tuning and Sizing Guide*

## Tuning Your Server for Performance

On the Performance Tuning page you can configure settings that control your Sun Java System Application Server's performance by controlling how many requests it can handle, how long requests remain open without activity before timing out, and whether you are doing reverse lookups of a client's IP using DNS. Also, if you are using DNS, you can set such performance-related features as whether you are using asynchronous DNS, and DNS caching settings.

For more information on tuning, see the *Sun Java System Application Server Performance Tuning Guide*.

To set the performance tuning settings:

1. In the left pane, click HTTP Server
2. Click the Tuning tab
3. Enter the desired value in the fields.
4. Click OK.

For additional information on the settings you can tune through the Administration interface, see the online help.

## Configuring HTTP Quality of Service

Quality of service refers to the performance limits you set for a server. For example, an ISP might want to charge different amounts of money for virtual servers depending on how much bandwidth allowed them.

Before you can use quality of service for a specific virtual server, you must enable it for the server instance and set some values.

To configure the quality of service settings for the server instance:

1. In the left pane, click HTTP Server.
2. Click the QOS tab.
3. To enable quality of service as a whole, click Enable.

By default quality of service is disabled. Enabling quality of service increases server overhead slightly.

4. Choose the Recompute Interval.

The recompute interval is the number of milliseconds between each computation of the bandwidth. The default is 100 milliseconds.

5. Choose the Metric Interval.

The metric interval is the interval in seconds during which the traffic is measured. The default is 30 seconds. All bandwidth measured during this time is averaged to give the bytes per second.

If your site has a lot of large file transfers, use a large value (several minutes or more) for this field. A large file transfer might take up all the allowed bandwidth for a short metric interval, and result in connections being denied if you've enforced the maximum bandwidth setting. Since the bandwidth is averaged by the metric interval, a longer interval smooths out spikes caused by large files.

If the bandwidth limit is much lower than available bandwidth (for example, 1 MB-per-second bandwidth limit but with a 1 GB-per-second connection to the backbone), the metric interval should be shortened.

Please note that if you have large static file transfers and a bandwidth limit that is much lower than available bandwidth, you have to decide which situation to tune for, since the problems require opposite solutions.

6. Set the bandwidth limit, in bytes per second, for the server.
7. Choose whether or not to enforce the bandwidth limit setting.

If you choose to enforce the bandwidth limit, once the server reaches its bandwidth limit additional connections are refused.

If you do not enforce the bandwidth limit, when the limit is exceeded the server logs a message to the error log.

8. Choose the maximum number of connections allowed for the server.

This number is the number of concurrent requests processed.

9. Choose whether or not to enforce the connection limit setting.

If you choose to enforce the connection limit, once the server reaches its limit additional connections are refused.

If you do not enforce the connection limit, when the limit is exceeded the server logs a message to the error log.

10. To specify additional name/value pairs, click the Properties button.
11. Click OK.

To configure quality of service using the command-line interface's `asadmin` utility, use the following commands:

- `create-http-qos`
- `delete-http-qos`

These commands use the following syntax:

```
asadmin create-http-qos --user admin_user [--password password] [--host hostname]
[--port admin_port] [--secure | -s] [--passwordfile file_name] [--virtualserver
virtual_server_id] [--bwlimit bandwidth_limit] [--enforcebwlimit enforce_bandwidth_limit]
[--connlimit connection_limit] [--enforceconnlimit enforce_connection_limit] instancename
```

```
asadmin delete-http-qos --user admin_user [--password password] [--host hostname]
[--port admin_port] [--secure | -s] [--passwordfile file_name] [--virtualserver
virtual_server_id] instancename
```

If you specify a virtual server, these commands create or delete quality of service information for that virtual server. If you do not specify a virtual server, the command affects the server instance.

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

For more information on the limitations to the quality of service features, see [“Administering the Transaction Service Using the CLI” on page 155.](#)

## Adding and Using Thread Pools

You can use thread pools to allocate a certain number of threads to a specific service, so that it doesn't take up more threads than you want it to. Another use for thread pools is for running thread-unsafe plug-ins. By defining a pool with the maximum number of threads set to 1, only one request is allowed into the specified service function.

When you add a thread pool, the information you specify includes the minimum and maximum number of threads, the stack size, and the queue size.

To add a thread pool:

1. In the left pane, click HTTP Server
2. Click Thread Pool.
3. Enter the desired value in the fields.
4. Click OK.

The thread pool is displayed at the bottom of the page. To edit or delete a thread pool, click the Edit or Delete button next to it.

After you've set up a thread pool, use it by designating it as the thread pool for a specific service.

For more information on using thread pools to improve performance, see the *Performance Tuning and Sizing Guide*.

## Editing Advanced Settings

When the Sun Java System Application Server starts up, it looks in a file called `init.conf` in the `instance_dir/config/` directory to establish a set of global variable settings that affect the server's behavior and configuration. Sun Java System Application Server executes all the directives defined in `init.conf`.

These settings are shown on the Advanced Settings page. You can edit certain settings in the `init.conf` file that affect the following areas:

- DNS
- SSL
- Performance
- CGI
- Keep-alive
- Logging

For a complete description of the `init.conf` file, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

To edit the advanced settings:

1. In the left pane, click HTTP Server
2. Click the Advanced tab
3. Click the type of settings you want to change (DNS, SSL, and so forth).
4. Make the desired changes to the settings and click OK.

For more information about each type of settings, see the online help.

## Configuring MIME Types

The Mime Types page allows you to edit your server's MIME files. MIME (Multi-purpose Internet Mail Extension) types control what types of multimedia files your system supports. MIME types also specify what file extensions belong to certain server file types, for example to designate what files are CGI programs.

You can create as many MIME types files as you need, and associate them with the application server instance or virtual servers. One MIME types file, `mime.types`, exists by default on the server, and cannot be deleted.

To create a new MIME Types file:

1. In the left pane, under HTTP Server, click MIME Type File.
2. On the right pane, click New.
3. Enter an identifier for the MIME file, and a file name.



4. Click OK.

To edit the definitions in a MIME file:

1. In the left pane, under HTTP Server, click the icon next to MIME Type File to expand the view.
2. Click the ID of the MIME file you want to edit.
3. On this page, edit the MIME file name associated with the ID.
4. To edit the file extensions in the MIME file, click Edit MIME file.
5. To edit an existing entry, click Edit next to it.
6. On the page that comes up, make your changes and click Change MIME Type.
7. To delete a MIME type, click Remove next to it.
8. To add a new MIME type, enter a category, content type, and file suffix in the fields, then click New Type.

To configure MIME types using the command-line interface's `asadmin` utility, use the following commands:

- `create-mime`
- `delete-mime`
- `list-mimes`

These commands use the following syntax:

```
asadmin create-mime --user admin_user [--password password] [--host hostname]
[--port admin_port] [--secure | -s] [--passwordfile file_name] [--instance
instancename] --mimefile filename mime_id
```

```
asadmin delete-mime --user admin_user [--password password] [--host hostname]
[--port admin_port] [--secure | -s] [--passwordfile file_name] [--instance
instancename] mime_id
```

```
asadmin list-mimes --user admin_user [--password password] [--host hostname]
[--port admin_port] [--secure | -s] [--passwordfile file_name] instancename
```

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

For information on using the MIME types with virtual servers, see the online help and [Chapter 15, “Using Virtual Servers.”](#)



# Using Virtual Servers

This chapter explains how to set up and administer virtual servers using your Sun Java System Application Server. For information on configuring settings for virtual server content, see [Chapter 16, “Managing Virtual Server Content.”](#)

This chapter includes the following topics:

- [Virtual Servers Overview](#)
- [Using Sun Java System Application Server Features with Virtual Servers](#)
- [Creating and Configuring HTTP Listeners](#)
- [Creating and Configuring Virtual Servers](#)
- [Deploying Virtual Servers](#)

## Virtual Servers Overview

When you use virtual servers you can offer companies or individuals domain names, IP addresses, and some server monitoring capabilities with a single installed server. For the users, it is almost as if they have their own web servers, though you provide the hardware and maintain the virtual servers.

When you install the unbundled version of the Sun Java System Application Server, a default virtual server for the application server instance is created. That is, for the default application server instance `server1`, a virtual server named `server1` is also created. If you are using the Solaris 9 bundled version, you need to create a server instance. When you create it, a virtual server with the same name is also created. A virtual server is created for each additional application server instance you create. For more information on creating and configuring virtual servers, see [“Creating and Configuring Virtual Servers” on page 374](#). For more information on deploying virtual servers, see [“Deploying Virtual Servers” on page 381](#).

This virtual server controls the Sun Java System Application Server's HTTP features that are available on a per-virtual server basis. You may not want to use multiple virtual servers, but you still configure certain properties for your application server instance by configuring the default virtual server created with that application server instance.

The settings for virtual servers are stored in the `virtual-server` element in the `server.xml` file, found in the `instance_dir/config` directory. If you change the machine's IP address for any reason, you must change the corresponding IP address in the `server.xml` file. For more information about this file, see *Sun Java System Application Server Administrator's Configuration File Reference*.

Some information pertaining to a virtual server is stored in its `obj.conf` file. Each virtual server has a separate `obj.conf` file.

This section includes the following topics:

- [HTTP Listeners](#)
- [Virtual Servers](#)
- [obj.conf File](#)
- [Virtual Server Selection for Request Processing](#)
- [Document Root](#)
- [Using Access Log Files and Server Log Files](#)

## HTTP Listeners

Connections between the server and clients happen on an HTTP listener, also called a listen socket. Each HTTP listener you create has an IP address, a port number, a return server name, and a default virtual server. If you want an HTTP listener to listen on all configured IP addresses on a given port for a machine, use `0.0.0.0`, `any`, `ANY`, or `INADDR_ANY` for the IP address. For more information on creating and configuring HTTP Listeners, see [“Creating and Configuring HTTP Listeners” on page 371](#).

When you install the unbundled version of Sun Java System Application Server, one HTTP listener, `http-listener-1`, is created automatically. This HTTP listener uses the IP address `0.0.0.0` and the port number you specified as your HTTP server port number during installation (the default is 80, or 1024 on UNIX if you are not installing as root). You cannot delete the default HTTP listener. If you are using multiple virtual servers, you can either use the default HTTP listener for all virtual servers, or create multiple HTTP listeners.

When you use the Solaris 9 bundled Sun Java System Application Server, your HTTP listener is created when you create the server instance. It has the IP address 0.0.0.0 and the port number you specified when you created your instance.

Since an HTTP listener is a combination of IP address and port number, you can have multiple HTTP listeners with the same IP address and different port numbers, or with different IP addresses and the same port number. For example, you could have 1.1.1.1:81 and 1.1.1.1:82. Additionally, you could have 1.1.1.1:81 and 1.2.3.4:81, as long as your machine is configured to respond to both these addresses. However, if you use the 0.0.0.0 IP address, which listens on all IP addresses on a port, you cannot set up HTTP listeners for additional IP addresses that listen on the same port for a specific IP address. For example, if you have an HTTP listener using 0.0.0.0:80 (all IP addresses on port 80) you cannot also create an HTTP listener which uses 1.2.3.4:80.

Each HTTP listener also has a default virtual server, which is the server to which it routes requests if it can't connect to the virtual server specified in the request.

In addition, you specify the number of acceptor threads (sometimes called accept threads) in the HTTP listener. Accept threads are threads that wait for connections. The threads accept connections and put them in a queue where they are then picked up by worker threads. Ideally, you want to have enough accept threads so that there is always one available when a new request comes in, but few enough so that they do not provide too much of a burden on the system. The default is 1. A good rule is to have one accept thread per CPU on your system. You can adjust this value if you find performance suffering.

You also specify whether security is enabled for an HTTP listener and what kind of security you are using (for example, what kind of SSL and which ciphers).

## Virtual Servers

To create a virtual server you must first decide what kind of virtual server you want. You can have an IP-address-based virtual server, or a URL-host-based virtual server. To create a virtual server, all you need to specify is a virtual server ID, one or more HTTP listeners, and one or more URL hosts.

This section includes the following topics:

- [Types of Virtual Servers](#)
- [IP-Address-Based Virtual Servers](#)
- [URL-Host-Based Virtual Servers](#)
- [Default Virtual Server](#)

## Types of Virtual Servers

All virtual servers have a URL host specified. However, the virtual server may also be associated with an IP address based on an HTTP listener. If the virtual server's HTTP listener listens on a specific IP address, the virtual server is called an IP-address-based virtual server.

If several virtual servers listen on the same IP address, they are distinguished by the URL host, and are URL-host-based virtual servers.

When a new request comes in, the server determines which virtual server to send it to based on the IP address or the value in the Host header. It evaluates the IP address first. For more information, see [“Virtual Server Selection for Request Processing” on page 368](#).

## IP-Address-Based Virtual Servers

In order to have multiple IP addresses on a single computer, you must either map them through the operating system or provide additional cards. To set up multiple IP addresses through the operating system, use the Network Control Panel (Windows) or the `ifconfig` utility (UNIX). Please note that directions for using `ifconfig` vary from platform to platform. Consult your operating system documentation for more information.

You create an IP-address-based virtual server by creating an HTTP listener that listens on a specific IP address. You then associate a virtual server as the default virtual server for the HTTP listener. For more information on ways to deploy virtual servers, see [“Deploying Virtual Servers” on page 381](#).

## URL-Host-Based Virtual Servers

You can set up URL-host-based virtual servers by giving them unique URL hosts. The contents of the Host request header directs the server to the correct virtual server.

For example, if you want to set up virtual servers for customers *aaa*, *bbb*, and *ccc* so that each customer can have an individual domain name, you first configure DNS to recognize that each customer's URL (`www.aaa.com`, `www.bbb.com`, `www.ccc.com`) resolves to the IP address of the HTTP listener you are using. You then set the URL hosts for each virtual server to the correct setting (for example, `www.aaa.com`). Note that you map hosts to IP addresses in the `/etc/hosts` file.

You can have any number of these URL-host-based virtual servers associated with an HTTP listener.

Because URL-Host-based virtual servers use the Host request header to direct the user to the correct page, not all client software works with them. Older client software that does not support the HTTP Host header won't work. These clients will receive the default virtual server for the HTTP listener.

## Default Virtual Server

URL-Host-based virtual servers are selected using the Host request header. If the end user's browser does not send the Host header, or if the server cannot find the specified Host header, the default virtual server for the HTTP listener services the request.

Also, for IP-address-based virtual servers, if Sun Java System Application Server cannot find the specified IP address, the default virtual server for the HTTP listener services the request. You can configure the default virtual server to send an error message, or server pages from a special document root.

---

<b>NOTE</b>	Do not confuse the default virtual server for an HTTP listener with the default virtual server created when you install the server. The default virtual server is the virtual server for the default application server instance. The default virtual server for an HTTP listener is any virtual server you designate as the default.
-------------	---

---

You specify a default virtual server when you create an HTTP listener. You can always change the default virtual server.

## obj.conf File

By default, each virtual server has a separate `obj.conf` file where virtual server settings are stored. When you change settings through the Administration interface or command-line interface, those changes are made automatically in the configuration files, including the virtual server's `obj.conf` file. All `obj.conf` files are located in the `instance_dir/config` directory. Whenever this guide refers to "the `obj.conf` file," it refers to all `obj.conf` files or to the `obj.conf` file for the virtual server being described.

The file named `obj.conf` that lacks a prefix is a template that Sun Java System Application Server uses to create `obj.conf` files for each virtual server. Editing this file does not affect any existing virtual servers, but does affect any subsequently created virtual servers. For more information on editing the `obj.conf` file directly, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

By default, each active `obj.conf` file is named *virtual\_server\_name-obj.conf*. Because the default virtual server for a server instance is named after the instance, when you first create a server instance, its `obj.conf` file is named *instance\_name-obj.conf*. Editing one of these files directly or through the Administration interface changes the configuration of a virtual server.

## Virtual Server Selection for Request Processing

Before the server can process a request, it must accept the request via an HTTP listener, then direct the request to the correct virtual server. This section discusses how the virtual server is determined.

- If the HTTP listener is configured to only a default virtual server, that virtual server is selected.
- If the HTTP listener has more than one virtual server configured to it, the request `Host` header is matched to the `hosts` attribute of a virtual server. If no `Host` header is present or no `hosts` attribute matches, the default virtual server for the HTTP listener is selected.

If a virtual server is configured to an SSL HTTP listener, its `hosts` attribute is checked against the subject pattern of the certificate at server startup, and a warning is generated and written to the server log if they don't match.

After the virtual server is determined, the Sun Java System Application Server executes the virtual server's `obj.conf` file. For details about how the server decides which directives to execute in `obj.conf`, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

## Document Root

The document root (sometimes called the primary document directory) is the central directory that contains all the virtual server's files you want to make available to remote clients.

The document root directory provides an easy way to restrict access to the files on a virtual server. It also makes it easy to move documents to a new directory (perhaps on a different disk) without changing any of the URLs because the paths specified in the URLs are relative to the primary document directory.



For example, if your document directory is *install\_dir/docs*, a request such as <http://www.sun.com/products/info.html> tells the server to look for the file in *install\_dir/docs/info.html*. If you change the document root (that is, you move all the files and subdirectories), you only have to change the document root that the virtual server uses, instead of mapping all URLs to the new directory or somehow telling clients to look in the new directory.

The document root for your default Sun Java System Application Server instance (server1) becomes the document root for the virtual servers created within the server1 application server instance. You can override that directory for each virtual server you create.

## Using Sun Java System Application Server Features with Virtual Servers

Sun Java System Application Server has many features, such as SSL and access control, that you can use with virtual servers. The following sections describe the features and provide information on where to look for more information.

This section includes the following topics:

- [Using SSL with Virtual Servers](#)
- [Using Access Log Files and Server Log Files](#)
- [Using Access Control with Virtual Servers](#)
- [Using CGIs with Virtual Servers](#)

### Using SSL with Virtual Servers

If you want to use SSL on a virtual server, in most cases you use an IP-address-based virtual server. The customary port is 443. It is difficult to use SSL on a URL-host-based virtual server because Sun Java System Application Server must read the request before determining which URL host to send the request to. Once the server reads the request, the initial handshake, where security information is exchanged, has already happened.

The only exception is when URL-Host-based virtual servers all have the same SSL configuration, including the same server certificate, using “wildcard certificates.” For more information, see the *Sun Java System Application Server Administrator’s Guide to Security*.

One way to implement SSL with virtual servers is to have two HTTP listeners, one using SSL and listening to port 443, and one that is not using SSL. A user would typically access the virtual server through the non-SSL HTTP listener. When the need to have secure transactions arises, users could click a button on the web page to start initiating secure transactions. After that, the requests go through the secure HTTP listener.

Because SSL transactions are much slower than non-SSL transactions, this design limits the SSL transactions to only the ones that are necessary. Faster, non-SSL connections are used the rest of the time.

For more information on setting up and using security with you Sun Java System Application Server and virtual servers, see the *Sun Java System Application Server Administrator's Guide to Security*. For a diagram of a sample SSL configuration with virtual servers, see [“Example 2: Secure Server” on page 383](#).

## Using Access Log Files and Server Log Files

The access log file is the file where HTTP accesses to the virtual server are logged. When you create a new virtual server, by default the access log file is the same log file as the application server instance. In many cases you will want each individual virtual server to have its own log file. To set this up, you can change the log path for each virtual server. If you want to keep all virtual server accesses logged to the same access log file, you can change the logging settings for the server instance so that the virtual server ID is included in the log file. For more information on changing the application server instance's logging, see [Chapter 5, “Using Logging.”](#)

The server log file is the file where informational messages and errors are logged. When you create a new virtual server, its log file by default is the same as the log file for the application server instance. You can change the log file for each virtual server.

## Using Access Control with Virtual Servers

With virtual servers you have the ability to set up access control on a per virtual server basis. You can even configure it so that each virtual server can have user and group authentication using an LDAP database. For more information, see the *Sun Java System Application Server Administrator's Guide to Security*.

## Using CGIs with Virtual Servers

You can use CGIs on virtual servers. You need to set up the directories where CGIs will be stored on each virtual server and set a file type for the CGI. For more information on CGIs, see the *Sun Java System Application Server Developer's Guide to Web Applications*.

## Creating and Configuring HTTP Listeners

Before the server can process a request, it must accept the request via an HTTP listener, then direct the request to the correct virtual server. One HTTP listener, `http-listener-1`, is created automatically when a server instance is created (either during installation or later). This HTTP listener uses the IP address 0.0.0.0 and the port number you specified as your application server port number. You cannot delete the default HTTP listener.

This section covers the following topics:

- [Creating an HTTP Listener](#)
- [Editing HTTP Listener Settings](#)
- [Deleting an HTTP Listener](#)

### Creating an HTTP Listener

To create an HTTP listener using the Administration interface:

1. In the left pane, for the application server instance, open HTTP Server.
2. Click HTTP Listeners.
3. Click New.
4. Fill in the fields.

HTTP listeners must have a unique combination of port number and IP address. You can use either IPV4 or IPV6 addresses. If you want to create an HTTP listener for IP-address-based virtual servers, specify a particular IP address at for the HTTP listener.

The Return Server Name field specifies the host name in the URLs the server sends to the client. This affects URLs the server automatically generates; it doesn't affect the URLs for directories and files stored in the server. This name should be the alias name if your server uses an alias.

The default virtual server is the virtual server that will answer requests for the HTTP listener if no other virtual server is found first. For more information, see [“Virtual Server Selection for Request Processing” on page 368](#).

You must enable the HTTP listener before it can accept requests.

You can also enable security and configure advanced properties for this HTTP listener. To specify IPV6, use the value `inet6` in the Family field. If this value is `inet6`, IPv4 addresses are prefixed with `::ffff:` in the server log.

## 5. Click OK.

Please note that you must enter an existing virtual server in the default virtual server field when you create an HTTP listener. You can use the virtual server created with the server instance, and then go back and change it after you've created additional virtual servers, if you like.

To create an HTTP listener using the command-line interface, use the `asadmin` utility's `create-http-listener` command. To get a list of all created HTTP listeners, use the command `list-http-listeners`.

To create an HTTP Listener, use the following syntax:

```
asadmin create-http-listener --user username [--password password] [--host
hostname] [--port adminport] [--secure | -s] [--passwordfile file_name] --address
address [--instance instancename] --listenerport listener_port --defaultvs virtual_server
--servername server_name [--family family] [--acceptorthreads acceptor_threads]
[--blockingenabled blocking_enabled] [--securityenabled security_enabled] [--enabled
enabled] listener_id
```

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Editing HTTP Listener Settings

To edit HTTP listener settings using the Administration interface:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open HTTP Listeners.

3. Click the HTTP listener you want to edit.
4. Make the desired changes and click Save.

For more information, see the online help.

You can also edit an HTTP listener using the `asadmin` utility in the command-line interface. Use the `get` command to get the current settings, and the `set` command to set them to new values.

To get the values of all the attributes of an HTTP listener:

```
asadmin> get server_instance.http-listener.http_listener_name.*
```

For example, to get the values for the default HTTP listener:

```
asadmin> get server1.http-listener.http-listener-1.*
```

To set the value of any attribute:

```
asadmin> set server_instance.http-listener.http_listener_name.attribute_name=value
```

For example, to set the attribute `defaultVirtualServer` to `server2` for `http-listener-1`:

```
asadmin> set
server1.http-listener.http-listener-1.defaultVirtualServer=server2
```

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Deleting an HTTP Listener

To delete an HTTP listener using the Administration interface:

1. In the left pane, for the application server instance, open HTTP Server.
2. Click HTTP Listeners.
3. Click the checkbox next to the HTTP listener you want to delete.
4. Click Delete.

To delete an HTTP listener using the command-line interface, use the `asadmin` utility's `delete-http-listener` command, using the following syntax:

```
asadmin delete-http-listener ---user username [--password password] [--host
hostname] [--port adminport] [--secure | -s] [--passwordfile file_name] --instance
instance httplistener_id
```

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Creating and Configuring Virtual Servers

Once you have set up an HTTP Listener, you can create and use virtual servers.

This section covers the following topics:

- [Creating a Virtual Server](#)
- [Editing Virtual Server Settings](#)
- [Deleting a Virtual Server](#)

### Creating a Virtual Server

To create a virtual server using the Administration interface:

1. In the left pane, for the application server instance, open HTTP Server.
2. Click Virtual Servers.
3. Click New.
4. Fill in the required fields and any optional fields.
5. Click Save.

To create a virtual server using the command-line interface, use the `asadmin` utility's `create-virtual-server` command, using the following syntax:

```
asadmin create-virtual-server --user username ---user username [--password
password] [--host hostname] [--port adminport] [--secure | -s] [--passwordfile
file_name] [--instance instancename] --hosts hosts --mime mime_types_file
[--httplisteners http-listeners] [--defaultwebmodule default_web_module]
[--configfile config_file] [--defaultobj default_object] [--state state] [--acls acls]
[--acceptlang accept_language] [--logfile logfile] [--property
(name=value) [ :name=value]*] virtual_server_id
```

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

When you create a virtual server, you can enter settings of the following types:

- [Required Settings](#)
- [Optional General Settings](#)
- [Web Application Settings](#)
- [CGI Settings](#)
- [HTTP Quality of Service Settings](#)

## Required Settings

The required setting for a virtual server include the name (ID), and the URL host or hosts.

You also must specify a MIME types file. The MIME types file contains the mappings of file extensions to types of files. For example, the MIME types file is where you can specify that all files ending `.cgi` be treated as CGI files.

You don't need to create a separate MIME types file for each virtual server. Instead, you create as many MIME types files as you need and associate them with a virtual server. The default MIME types file is called `mime1` and the file name is `mime.types`.

For more information on MIME types files, see [“Configuring MIME Types” on page 360](#).

## Optional General Settings

In addition to the required fields, you can also set optional fields.

### *HTTP listener*

The HTTP listener handles the connection to the virtual server. You must specify one in order to have remote clients access the virtual server.

### *ACLs*

The access control list (ACL) applied to the virtual server. For more information, see the *Sun Java System Application Server Administrator's Guide to Security*.

### *Accept Language Header*

When clients contact a server using HTTP 1.1, they can send header information describing the languages they accept. You can configure your server to parse this language information.

For example, if you store documents in Japanese and English, you could choose to parse the accept language header. When clients that have Japanese as the accept language header contact the server, they receive the Japanese version of the page. When clients that have English as the accept language header contact the server, they receive the English version.

If you do not support multiple languages, you should not parse the accept language header.

### *State*

This state is the virtual server's state, which is independent of whether the application server instance is On or Off. If a virtual server's state displayed on this page is On, the virtual server can only accept requests if the application server instance is On as well.

This is true of the default virtual server for the default application server instance as well. If you turn off your application server instance, your default virtual server is still set to On, but will not accept connections.

Valid states are On, Off, or Disabled. A virtual server is able to accept connections if it is set to On

You cannot turn off or disable the default virtual server for the application server instance.

### *Log Files*

The log file, (also known as the server log file) is the file where informational messages and errors are logged. The access log file is the file where HTTP accesses to the virtual server are logged.

### *Document Root*

The document root (sometimes called the primary document directory) is the central directory that contains all the virtual server's files you want to make available to remote clients. For more information, see [“Document Root” on page 368](#).

## Web Application Settings

A web application is a collection of servlets, JavaServer Pages, HTML documents, and other web resources which might include image files, compressed archives, and other data. A web application may be packaged into an archive (a WAR file) or exist in an open directory structure.



Sun Java System Application Server 7 supports the Servlet 2.3 API specification, which allows servlets and JSPs to be included in web applications. In addition, Sun Java System Application Server 7 supports SHTML and CGI, which are non-J2EE application components.

When you create a virtual server, you specify a default web module for the virtual server. The default web module responds to all requests that cannot be resolved to other web modules deployed to the virtual server. If you don't specify a default web module, the web module that has an empty context root is used. If there's no web module with an empty context root, a system default web module is created and used.

When you deploy a web application, you specify a virtual server. Once you have deployed a web application, it appears in the list of available web modules to choose as the default web module for a virtual server. When you specify a web module as the default web module for a virtual server, the virtual server is automatically added to the web application's list of virtual servers.

## CGI Settings

The CGI settings you set when you create a virtual server govern the user and group CGI programs run as, the directory to change to (chroot) before CGI execution begins, and the directory to change to after the chroot.

On UNIX you can also set nice, an increment that determines a CGI program's priority relative to the server. Typically, the server is run with a nice value of 0 and the nice increment would be between 0 (the CGI program runs at same priority as server) and 19 (the CGI program runs at much lower priority than server).

## HTTP Quality of Service Settings

Quality of service refers to the performance limits you set for a virtual server. For example, an ISP might want to charge different amounts of money for virtual servers depending on how much bandwidth allowed them. These settings can either be enforced (that is, only the specified bandwidth and maximum number of connections will be allowed) or not enforced. If the settings are not enforced, a message is logged to the log file when the limits are exceeded. For more information, see [“Administering the Transaction Service Using the CLI” on page 155](#).

In addition to changing these settings through the Administration interface, you can use the command-line interface's `asadmin` utility. To configure quality of service using the command-line interface's `asadmin` utility, use the following commands:

- `create-http-qos`
- `delete-http-qos`

These commands use the following syntax:

```
asadmin create-http-qos --user username [--password password] [--host hostname]
[--port adminport] [--secure | -s] [--passwordfile file_name] [--virtualserver
virtual_server_id] [--bwlimit bandwidth_limit] [--enforcebwlimit enforce_bandwidth_limit]
[--connlimit connection_limit] [--enforceconnlimit enforce_connection_limit] instance_name

asadmin delete-http-qos --user username [--password password] [--host hostname]
[--port adminport] [--secure | -s] [--passwordfile file_name] [--virtualserver
virtual_server_id] instance_name
```

If you specify a virtual server, these commands create or delete quality of service information for that virtual server. If you do not specify a virtual server, the command affects the server instance.

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Editing Virtual Server Settings

Once you have set up your virtual servers, you can edit them. For information on editing virtual server settings, see the following topics:

- [Editing General Settings Using the Administration Interface](#)
- [Editing General Settings Using the Command-Line Interface](#)
- [Editing CGI Settings](#)
- [Editing Document Handling Settings, Document Directories Settings, and HTTP/HTML Settings](#)

### Editing General Settings Using the Administration Interface

The virtual server's general settings are the ones you could set when you created the virtual server. To change them, follow these steps:

1. In the left pane, for the application server instance, open HTTP Server.

2. Open Virtual Servers.
3. Click the virtual server you want to edit.
4. Make your desired changes.

The areas you can change include quality of service settings, adding ACLs, content-related settings such as the document root and the accept language header, CGI-related settings such as the user, group, nice, and chroot settings, and the default web module.

5. Click Save.

For more information on some of these settings, see [“Creating and Configuring Virtual Servers” on page 374](#). Also see the online help.

## Editing General Settings Using the Command-Line Interface

You can also edit these setting using the `asadmin` utility in the command-line interface. Use the `get` command to get the current settings, and the `set` command to set them to new values.

To get all the attributes from a virtual server, use the following syntax:

```
asadmin> get instance_name.virtual-server.vserver_id.*
```

For example:

```
asadmin> get server1.virtual-server.vsl.*
```

If you want to get all the attributes for the application server instance `server1`, use the following syntax:

```
asadmin> get server1.virtual-server.server1.*
```

To set an attribute, for example, the accept language header, use the following syntax:

```
asadmin> set
server1.virtual-server.server1.virtualserver.acceptLanguage=false
```

---

**NOTE** You can use the command-line interface to set values for all the fields on the General page. However, you cannot use the command-line interface to set values for the fields on the pages on other tabs, for example, the pages on the CGI tab.

---

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

## Editing CGI Settings

For information on editing CGIs, see the *Sun Java System Application Server Developers Guide to Web Applications*.

## Editing Document Handling Settings, Document Directories Settings, and HTTP/HTML Settings

For information on changing these settings, see [Chapter 16, “Managing Virtual Server Content.”](#)

# Deleting a Virtual Server

To delete a virtual server:

1. In the Administration interface, in the left pane, for the application server instance, open HTTP Server.
2. Click Virtual Servers.
3. Click the checkbox next to the virtual server you want to delete.
4. Click Delete.

You cannot delete all virtual servers using the Administration interface.

To delete a virtual server using the command-line interface, use the `asadmin` utility's `delete-virtual-server` command.

Usage is as follows:

```
asadmin delete-virtual-server --user username [--password password] [--host hostname] [--port adminport] [--secure | -s] [--passwordfile file_name] --instance instance virtualserver_id
```

For more information on command syntax, see the command-line interface help. For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

# Deploying Virtual Servers

Sun Java System Application Server's virtual server architecture is very flexible. An application server instance can have any number of HTTP listeners, both secure and non-secure. You can associate any number of virtual servers with these HTTP listeners. You can have both IP-address-based and URL-host-based virtual servers.

Every virtual server can (but does not have to) have its own list of ACLs, its own `mime.types` file, and its own set of Java Web Applications.

This design gives you maximum flexibility to configure the server for a variety of applications. The following examples discuss some of the possible configurations available for Sun Java System Application Server.

- [Example 1: Default Configuration](#)
- [Example 2: Secure Server](#)
- [Example 3: Intranet Hosting](#)
- [Example 4: Mass Hosting](#)

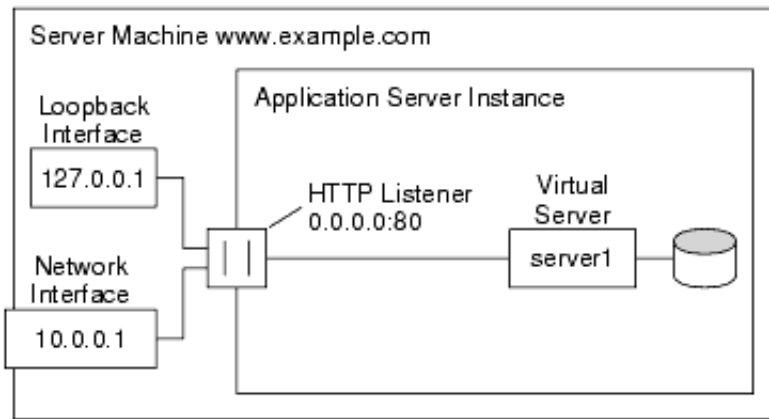
## Example 1: Default Configuration

The default configuration is one application server instance. This application server instance has just one HTTP listener listening on port 80, 1024, or whatever you selected, of any IP address to which your computer is configured.

Some mechanism in your local network establishes a name-to-address mapping for each of the addresses to which your computer is configured. In the following example, the computer has two network interfaces: the loopback interface (the interface that exists even without a network card) on address 127.0.0.1, and an ethernet interface on address 10.0.0.1.

The name `example.com` is mapped to 10.0.0.1 via DNS. The HTTP listener is configured to listen on port 80 on any address to which that machine is configured ("`0.0.0.0:80`").

As there are no IP-address-based virtual servers in the default configuration, the only HTTP listener is the default one. All connections pass through to virtual server `server1`.

**Figure 15-1** Default Configuration**DNS**

www.example.com	10.0.0.1

In this configuration, connections to the following reach the server and are served by virtual server VS1

- `http://127.0.0.1/` (initiated on example.com)
- `http://localhost/` (initiated on example.com)
- `http://example.com/`
- `http://10.0.0.1/`

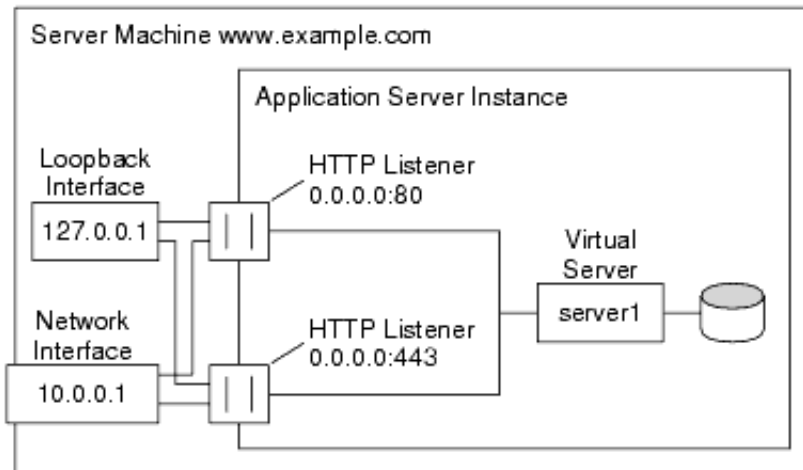
Use this configuration for traditional HTTP server use. You do not need to add additional virtual servers or HTTP listeners. You configure the settings of the server by changing the settings for server1.

## Example 2: Secure Server

If you want to use SSL in the default configuration, you can simply change the HTTP listener to secure mode.

You can also add a new secure HTTP listener configured to 0.0.0.0:443 and associate server1 to the new HTTP listener. The virtual server now has HTTP listeners, one that uses the secure HTTP listener, and one that doesn't. Now your server will serve the same content both with and without SSL, i.e. `http://example.com/` and `https://example.com/` deliver the same content.

**Figure 15-2** Secure Server



### DNS

www.example.com	10.0.0.1

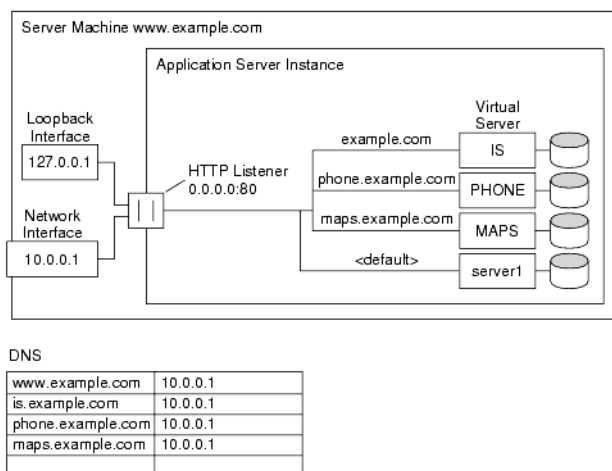
Please note that the SSL parameters are attached to the HTTP listener.

## Example 3: Intranet Hosting

A more complex configuration of the Sun Java System Application Server is one in which the server hosts a few virtual servers for an intranet deployment. For example, you have three internal sites where employees can look up other users' phone numbers, look at maps of the campus, and track the status of their requests to the Information Services department. Previously (in this example), these sites were hosted on three different computers that had the names `phone.example.com`, `maps.example.com` and `is.example.com` mapped to them.

To minimize hardware and administrative overhead, you want to consolidate all three sites into one application server living on the machine `example.com`. You could set this up in two ways: using URL-host-based or IP-address-based virtual servers. Both have distinct advantages and disadvantages.

**Figure 15-3** Intranet Hosting Using URL-Host-Based Virtual Servers



While URL-host-based virtual servers are easy to set up, they have the following disadvantages:

- Supporting SSL in this configuration requires non-standard setup using wildcard certificates. For more information see the *Sun Java System Application Server Administrator's Guide to Security*.
- URL-host-based virtual servers don't work with legacy HTTP clients



The advantages to IP-address-based virtual servers are:

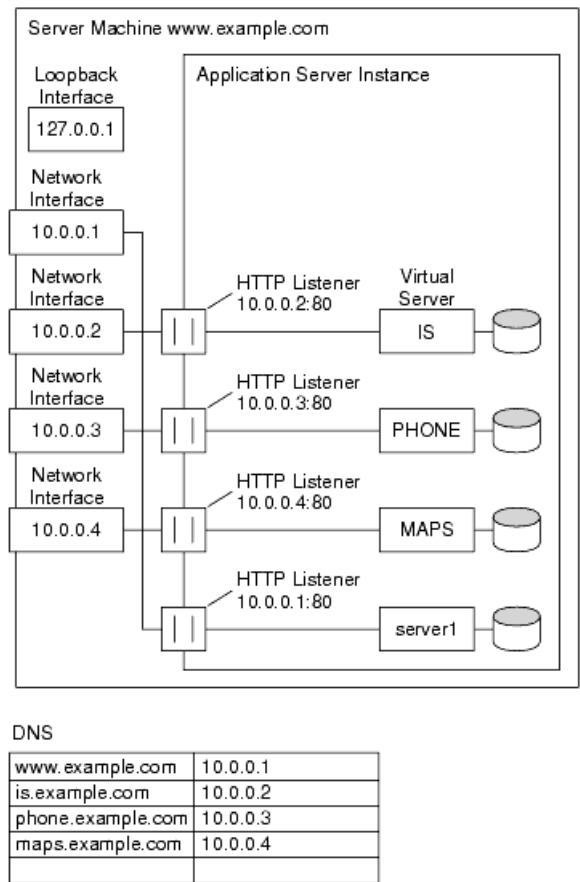
- They work with older clients that do not support the HTTP/1.1 Host header.
- Providing SSL support is straightforward.

The disadvantages are:

- They require configuration changes on the host computer (configuration of real or virtual network interfaces)
- They don't scale to configurations with thousands of virtual servers

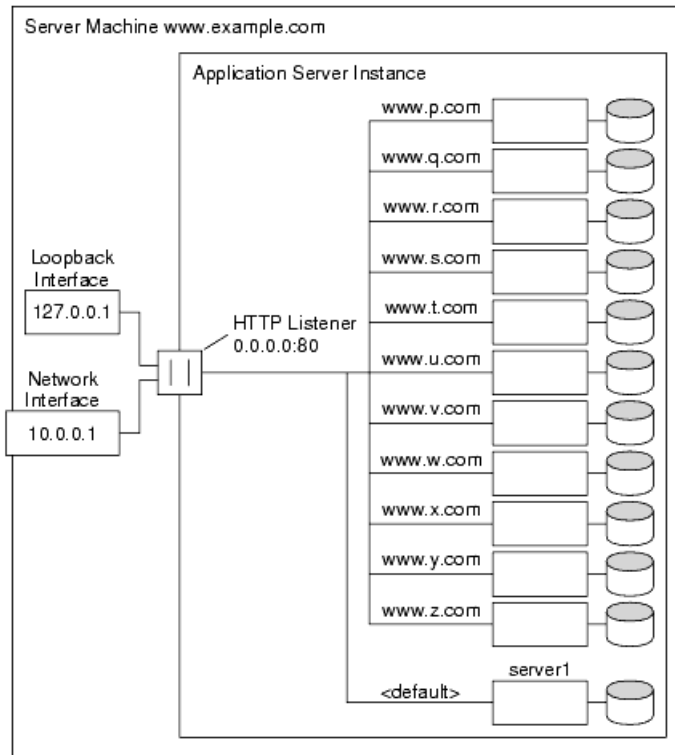
Both configurations require setting up name-to-address mappings for the three names. In the IP-address-based configuration, each name maps to a different address. The host machine must be set up to receive connections on all these addresses. In the URL-host-based configuration, all names can map to the same address, the one the machine had originally.

**Figure 15-4** Intranet Hosting Using IP-Address-Based Virtual Servers



## Example 4: Mass Hosting

Mass hosting is a configuration in which you enable many low-traffic virtual servers. For example, an ISP that hosts many low-traffic personal home pages would fall into this category. The virtual servers are usually URL-host-based.

**Figure 15-5** Mass Hosting**DNS**

www.example.com	10.0.0.1
www.p.com	10.0.0.1
www.q.com	10.0.0.1
www.r.com	10.0.0.1
...	
www.z.com	10.0.0.1

Notice that the default virtual server, `server1`, still exists.



# Managing Virtual Server Content

This chapter describes how you can configure and manage the files served by virtual servers.

This chapter includes the following topics:

- [Changing the Document Root](#)
- [Setting Additional Document Directories](#)
- [Enabling Remote File Manipulation](#)
- [Using htaccess](#)
- [Restricting Symbolic Links \(UNIX\)](#)
- [Customizing User Public Information Directories \(UNIX\)](#)
- [Setting the Document Preferences](#)
- [Customizing Error Responses](#)
- [Changing the International Character Set](#)
- [Setting the Document Footer](#)
- [Configuring URL Forwarding](#)
- [Setting up Server-Parsed HTML](#)
- [Setting Cache Control Directives](#)
- [Using Stronger Ciphers](#)

# Changing the Document Root

The document root is the central directory where you store all the files you want to make available to remote clients.

When you add a virtual server, you specify a document root with an absolute path. For more information about the document root and how it is used, see [“Document Root” on page 368](#).

To use the Administration interface to change the document root to use a different path:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the General Tab.
5. Enter an absolute directory path in the Document Root field.

You need to create this directory manually.

6. Click OK.

For more information, see the online help.

---

**NOTE** Typically, each virtual server has its own document root.

---

# Setting Additional Document Directories

Most of the time, the documents for a virtual or server instance are in the document root. Sometimes, though, you may want to serve documents from a directory outside of the document root. You can do this by setting additional document directories. By serving from a document directory outside of the document root, you can let someone manage a group of documents without giving them access to your primary document root.

To use the Administration interface to add an additional document directory:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.

4. Click the Doc Directories tab.
5. Click Additional Doc Directories.
6. Choose the URL prefix to map.

Clients send this URL to the server when they want documents.

7. Specify the directory to map those URLs to.
8. Click OK.

To for more information, see the online help.

You should restrict access to additional document directories so that users cannot write to them.

## Enabling Remote File Manipulation

When you enable remote file manipulation, clients are able to upload files, delete files, create directories, remove directories, list the contents of a directory, and rename files on your server. The virtual servers' configuration file `obj.conf` contains the commands that are activated when you enable remote file manipulation. By activating these commands, you allow remote browsers to change a server's documents. You should use access control to restrict write access to these resources to prevent unauthorized tampering.

Note that enabling remote file manipulations should have no effect on using content management systems such as Microsoft Frontpage.

**UNIX:** You must have the correct permissions for your files or this function will not work; that is, the document root user must be the same as the server user.

To use the Administration interface to enable remote file manipulation:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the Doc Directories tab.
5. Click Remote File Manipulation.
6. Choose Entire Server from the resource picker to apply your change to the whole virtual server, or navigate to a specific directory within a virtual server.
7. Choose to activate remote file manipulation.

8. Click OK.

For more information, see the online help.

## Using htaccess

The `htaccess` files are dynamic configuration files that store a subset of configuration options. You can use `htaccess` files in combination with the Application Server standard access controls (standard access controls are always applied before any `htaccess` access controls).

For information on using `htaccess`, see the *Sun Java System Application Server Administrator's Guide to Security*.

## Restricting Symbolic Links (UNIX)

You can limit the use of the file system links in your server. File system links are references to files stored in other directories or file systems. The reference makes the remote file as accessible as if it were in the current directory. There are two types of file system links:

- Hard links—A hard link is really two filenames that point to the same set of data blocks; the original file and the link are identical. For this reason, hard links cannot be on different file systems.
- Symbolic (soft) links—A symbolic link consists of two files, an original file that contains the data, and another that points to the original file. Symbolic links are more flexible than hard links. Symbolic links can be used across different file systems and can be linked to directories.

For more information about hard and symbolic links, see your UNIX system documentation.

File system links are an easy way to create pointers to documents outside of the primary document directory and anyone can create these links. For this reason you might be concerned that people might create pointers to sensitive files (for example, confidential documents or system password files).

To use the Administration interface to restrict symbolic links:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.



3. Click the name of the virtual server you want to edit.
4. Click the Doc Directories tab.
5. Click Symbolic Links.
6. Choose Entire Server from the resource picker to apply your change to the whole virtual server, or navigate to a specific directory within a virtual server.
7. Choose whether to enable soft and/or hard links and the directory to start from.
8. Click OK.

For more information, see the online help.

## Customizing User Public Information Directories (UNIX)

Sometimes users want to maintain their own web pages. You can configure public information directories that let all the users on a server create home pages and other documents without your intervention.

With this system, clients can access your server with a certain URL that the server recognizes as a public information directory. For example, suppose you choose the prefix `~` and the directory `public_html`. If a request comes in for `http://www.sun.com/~jdoe/aboutjane.html`, the server recognizes that `~jdoe` refers to a users' public information directory. It looks up `jdoe` in the system's user database and finds Jane's home directory. The server then looks at `~/jdoe/public_html/aboutjane.html`.

This section contains the following topics:

- [Configuring Public Information Directories](#)
- [Restricting Content Publication](#)
- [Loading the Entire Password File on Startup](#)

## Configuring Public Information Directories

To use the Administration interface to configure your virtual server to use public directories:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the Doc Handling tab.
5. Click User Doc Directories.
6. Choose a user URL prefix.

The usual prefix is `~` because the tilde character is the standard UNIX prefix for accessing a user's home directory.

7. Choose the subdirectory in the user's home directory where the server looks for HTML files.

A typical directory is `public_html`.

8. Designate the password file.

The server needs to know where to look for a file that lists users on your system. The server uses this file to determine valid user names and to find their home directories. If you use the system password file for this purpose, the server uses standard library calls to look up users. Alternatively, you can create another user file to look up users. You can specify that user file with an absolute path.

Each line in the file should have this structure (the elements in the `/etc/passwd` file that aren't needed are indicated with `*`):

```
username:***:groupid:*:homedir:*
```

9. Choose whether to load the password database at startup.

For more information, see [“Loading the Entire Password File on Startup” on page 395](#).

10. Click OK.

For more information, see the online help.

Another way to give users separate directories is to create a URL mapping to a central directory that all of your users can modify.

## Restricting Content Publication

In some situations a system administrator may want to restrict what user accounts are able to publish content via user document directories. To restrict a user's publishing, add a trailing slash to the user's home directory path in the `/etc/passwd` file:

```
jdoe::1234:1234:John Doe:/home/jdoe:/bin/sh
```

becomes:

```
jdoe::1234:1234:John Doe:/home/jdoe/:/bin/sh
```

After you make this modification, Sun Java System Application Server will not serve pages from this user's directory. The browser requesting the URI receives a "404 File Not Found" error and a 404 error will be logged to the access log.

If, at a later time, you decide to allow this user to publish content, remove the trailing slash from the `/etc/passwd` entry, then restart the Application Server Instance.

## Loading the Entire Password File on Startup

You also have the option of loading the entire password file on startup. If you choose this option, the server loads the password file into memory when it starts, making user lookups much faster. If you have a very large password file, however, this option can use too much memory.

## Setting the Document Preferences

This section contains the following topics:

- [Entering an Index Filename](#)
- [Selecting Directory Indexing](#)
- [Specifying a Server Home Page](#)
- [Specifying a Default MIME Type](#)

To use the Administration interface to set the document preferences, follow these steps:

1. In the left pane, for the application server instance, open HTTP Server.

2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the Doc Handling tab.
5. Click Doc Preferences.
6. Choose the appropriate field values, as discussed in the following sections.
7. Click OK.

The preferences you can set are discussed more fully in the sections that follow. For additional information, see the online help.

## Entering an Index Filename

If a document name is not specified in the URL the server automatically displays the index file. The default index files are `index.html` and `home.html`. If more than one index file is specified, the server looks in the order in which the names appear in this field until one is found. For example, if your index filenames are `index.html` and `home.html`, the server looks for `index.html` and if it doesn't find it looks for `home.html`.

## Selecting Directory Indexing

A document directory will probably have several subdirectories. For example, there might be a directory called `products`, another called `people`, and so on. It's often helpful to let clients access an overview (or index) of these directories.

The server indexes directories by searching the directory for an index file called `index.html` or `home.html`, which is a file you create and maintain as an overview of the directory's contents. For more information, see [“Entering an Index Filename” on page 396](#). You can specify any file as an index file for a directory by naming it one of these default names, which means you can also use a CGI program as an index.

If an index file isn't found, the server generates an index file that lists all the files in the document root.

---

**CAUTION** If your server is outside the firewall, turn off directory indexing to ensure that your directory structure and filenames are not accessible.

---

## Specifying a Server Home Page

When end users first access the server, the first file they see is usually called a home page. Usually, this file has general information about your server and links to other documents.

By default, the server finds the index file specified in the Index Filename field in the Document Preferences page and uses that for the home page. However, you can also specify a file to use as the home page.

## Specifying a Default MIME Type

When a document is sent to a client, the server includes a section that identifies the document's type, so the client can present the document in the right way. However, sometimes the server can't determine the proper type for the document because the document's extension is not defined for the server. In those cases, a default value is sent.

The default is usually text/plain, but you should set it to the type of file most commonly stored on your server. Some common MIME types include the following:

- text/plain
- text/richtext
- image/jpeg
- application/x-tar
- application/x-gzip
- text/html
- image/tiff
- image/gif
- application/postscript
- audio/basic

## Customizing Error Responses

You can specify a custom error response that sends a detailed message to clients when they encounter errors from your virtual server. You can specify a file to send or a CGI program to run.

For example, you can change the way the server behaves when it gets an error for a specific directory. If a client tries to connect to a part of your server protected by access control, you might return an error file with information on how to get an account.

Before you can enable a custom error response, you must create the HTML file to send or the CGI program to run in response to an error. After you do this, enable the response in the Administration interface.

To use the Administration interface to enable a customized error response:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the Doc Handling tab.
5. Click Error Responses.
6. Choose Entire Server from the resource picker to apply your change to the whole virtual server, or navigate to a specific directory within a virtual server.
7. For each error code you want to change, specify the absolute path to the file or CGI that contains the error response.
8. Click OK.

For more information see the online help.

## Changing the International Character Set

The character set of a document is determined in part by the language it is written in. You can override a client's default character set setting for a document, a set of documents, or a directory by selecting a resource and entering a character set for that resource.

Browsers can use the MIME type `charset` parameter in HTTP to change its character set. If the server includes this parameter in its response, the browser changes its character set accordingly. Examples are:

- `Content-Type: text/html; charset=iso-8859-1`
- `Content-Type: text/html; charset=iso-2022-jp`

The following `charset` names are specified in RFC 1700 (except for the names that begin with `x-`):

- `us-ascii`
- `iso-8859-1`
- `iso-2022-jp`
- `x-sjis`
- `x-euc-jp`
- `x-mac-roman`

Additionally, the following aliases are recognized for `us-ascii`:

- `ansi_x3.4-1968`
- `iso-ir-6`
- `ansi_x3.4-1986`
- `iso_646.irv:1991`
- `ascii`
- `iso646-us`
- `us`
- `ibm367`
- `cp367`

The following aliases are recognized for `iso_8859-1`:

- `latin1`
- `iso_8859-1`
- `iso_8859-1:1987`
- `iso-ir-100`
- `ibm819`
- `cp819`

To use the Administration interface to change the character set:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the Doc Handling tab.
5. Click International Characters.

6. Choose Entire Server from the resource picker to apply your change to the whole virtual server, or navigate to a specific directory within a virtual server.
7. Set the character set for all or part of the server.

If you leave this field blank, the character set is set to NONE.

8. Click OK.

For more information, see the online help.

## Setting the Document Footer

You can specify a document footer, which can include the last-modified time, for all the documents in a certain section of the server. This footer works for all files except output of CGI scripts or parsed HTML (.shtml) files. If you need your document footer to appear on CGI-script output or parsed HTML files, enter your footer text into a separate file and add a line of code or another server-side include to append that file to the page's output.

To use the Administration interface to set the document footer, follow these steps:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the Doc Handling tab.
5. Click Doc Footer.
6. Choose Entire Server from the resource picker to apply your change to the whole virtual server, or navigate to a specific directory within a virtual server.

If you choose a directory, the document footer applies only when the server receives a URL for that directory or any file in that directory.

7. Specify the type of files that you want to have include the footer.
8. Specify the date format.
9. Type any text you want to have appear in the footer.

The maximum number of characters for a document footer is 765. If you want to include the date the document was last modified, type the string :LASTMOD:.

For more information see the online help.



# Configuring URL Forwarding

URL forwarding allows you to redirect document requests to another server. Forwarding URLs or redirection is a method for the server to tell a user that a URL has changed (for example, because you have moved files to another directory or server). You can also use redirection to seamlessly send a person who requests a document on one server to a document on another server.

For example, if you forward `http://www.sun.com/info/movies` to a prefix `film.sun.com`, the URL `http://www.sun.com/info/movies` redirects to `http://film.sun.com/info/movies`.

Sometimes you may want to redirect requests for all the documents in one sub-directory to a specific URL. For example, if you had to remove a directory because it was causing too much traffic, or because the documents were no longer to be served for any reason, you could direct a request for any one the documents to a page explaining why the documents were no longer available. For example, a prefix on `/info/movies` could be redirected to `http://www.sun.com/explain.html`.

To use the Administration interface to configure URL forwarding:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the HTTP/HTML tab.
5. Click URL Forwarding.
6. Type the URL prefix you want to redirect, and whether you want to redirect it to another prefix or to a static URL.
7. Click OK.

For more information see the online help.

# Setting up Server-Parsed HTML

HTML is normally sent to the client exactly as it exists on disk without any server intervention. However, the server can search HTML files for special commands (that is, it can parse the HTML) before sending documents. If you want the server to parse these files and insert request-specific information or files into documents, you must first enable HTML parsing.

To use the Administration interface to set up HTML parsing:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the HTTP/HTML tab.
5. Click Parse HTML.
6. Choose Entire Server from the resource picker to apply your change to the whole virtual server, or navigate to a specific directory within a virtual server.

If you choose a directory, the server will parse HTML only when the server receives a URL for that directory or any file in that directory.

7. Choose whether to activate server-parsed HTML.

You can activate for HTML file s but not the exec tag, or for HTML files and the exec tag, which allows HTML files to execute other programs on the server.

8. Choose which files to parse.

You can choose whether to parse only files with the .shtml extension, or all HTML files, which slows performance. If you are using UNIX, you can also choose to parse UNIX files with the execute permission turned on, though that can be unreliable.

9. Click OK.

For more information on setting your server to accept parsed HTML, see the online help.

For more information on using server-parsed HTML, see the *Sun Java System Application Server Developer's Guide to Web Applications*.

## Setting Cache Control Directives

Cache-control directives are a way for Sun Java System Application Server to control what information is cached by a proxy server. Using cache-control directives, you override the default caching of the proxy to protect sensitive information from being cached, and perhaps retrieved later. For these directives to work, the proxy server must comply with HTTP 1.1.

For more information HTTP 1.1, see the Hypertext Transfer Protocol--HTTP/1.1 specification (RFC 2068) at:

<http://www.ietf.org/>

To use the Administration interface to set cache control directives:

1. In the left pane, for the application server instance, open HTTP Server.
2. Open Virtual Servers.
3. Click the name of the virtual server you want to edit.
4. Click the HTTP/HTML tab.
5. Click Cache Control Directives
6. Fill in the fields. Valid values for the response directives are as follows:
  - **Public.** The response is cachable by any cache. This is the default.
  - **Private.** The response is only cachable by a private (non-shared) cache.
  - **No Cache.** The response must not be cached anywhere.
  - **No Store.** The cache must not store the request or response anywhere in nonvolatile storage.
  - **Must Revalidate.** The cache entry must be revalidated from the originating server.
  - **Maximum Age (sec).** The client does not accept a response that has an age greater than this age.
7. Click OK.

For more information see the online help.

## Using Stronger Ciphers

For information on setting stronger ciphers, see the *Sun Java System Application Server Administrator's Guide to Security*.



# Configuring Multiple Server Instances

Chapter 17, “Configuring HTTP Load Balancing and Failover (Enterprise Edition)”

Chapter 18, “Configuring RMI/IIOP Load Balancing and Failover (Enterprise Edition)”

Chapter 19, “Managing Clusters (Enterprise Edition)”

Chapter 20, “Configuring Session Persistence (Enterprise Edition)”

Chapter 21, “Administering the High-Availability Database (Enterprise Edition)”



# Configuring HTTP Load Balancing and Failover (Enterprise Edition)

This chapter describes how to setup load balancing and failover of HTTP RMI-IIOP requests.

This chapter includes the following topics:

- [About Load Balancing and Failover](#)
- [Configuring HTTP Load Balancing and Failover](#)
- [Monitoring the Load Balancer Plug-in](#)
- [Monitoring the Load Balancer Plug-in](#)
- [Known Issues in Load Balancing Requests](#)

## About Load Balancing and Failover

Sun™ Java™ System Application Server Enterprise Edition provides high availability of HTTP sessions. The goal of load balancing is to evenly distribute the workload between multiple Sun Java System Application Server instances, thereby increasing the overall throughput of the system.

This section discusses the following topics:

- [Requirements for Load Balancing](#)
- [Using Third Party Load Balancers](#)
- [Understanding Assigned Requests and Unassigned Requests](#)
- [HTTP Load Balancing Algorithm](#)

## Requirements for Load Balancing

The following sections lists the requirements for load balancing HTTP requests.

You must meet the following requirements before you can use the load balancer plug-in:

- Sun Java System Application Server Enterprise Edition is installed.
- A web server is installed (Sun Java System Web Server 6.1 or Apache 1.23.29).
- At least two application server instances are created and HTTP listeners configured in a cluster.
- Deployed your web applications to all application server instances that will participate in load balancing.

---

**NOTE**

- The HTTP listeners and RMI/IIOP endpoints (if you are using RMI/IIOP load balancing) must be configured on the same cluster of Application Server instances.
  - For more information on clusters, see [Chapter 19, “Managing Clusters \(Enterprise Edition\).”](#)
  - For more information on RMI/IIOP load balancing, see [Chapter 18, “Configuring RMI/IIOP Load Balancing and Failover \(Enterprise Edition\).”](#)
- 

## Using Third Party Load Balancers

If you use a third party software or hardware load balancer instead of the bundled load balancer plug-in (between the web server and application server), you should configure the third party load balancer to support session affinity.

However, if the third party load balancer is configured to load balance requests to multiple web servers (the bundled load balancer plug-in continues to communicate between the web server and Application Server), there is no need to configure session affinity for the third party load balancer. In such a scenario, session affinity is managed by the bundled load balancer plug-in.

For a list of supported third party load balancers, see the Release Notes at the product's documentation web site.



## Understanding Assigned Requests and Unassigned Requests

When a request first comes in from a HTTP client to the load balancer, it is a request for a new session. A request for a new session is called an *unassigned* request. The load balancer routes this request to an application server instance in the cluster according to a round-robin algorithm. For more information, see [“HTTP Load Balancing Algorithm” on page 409](#).

Once a session is created on an application server instance, the load balancer routes all subsequent requests for this session only to that particular instance. A request for an existing session is called an *assigned* or a *sticky* request.

## HTTP Load Balancing Algorithm

The Sun Java System Application Server load balancer uses a sticky round robin algorithm to load balance incoming HTTP and HTTPS requests. This means that all requests for a given session are sent to the same application server instance. With a sticky load balancer, the session data can be cached on a single application server rather than having to be distributed to all instances in a cluster.

Therefore, the sticky round robin scheme provides significant performance benefits that normally override the benefits of a more evenly distributed load that can be obtained with pure round robin.

### About Sticky Round Robin Load Balancing Algorithm

When a new HTTP request is sent to the load balancer plug-in, it's forwarded to an application server instance based on a simple round robin scheme. Subsequently, this request is “stuck” to this particular application server instance, either by using cookies, or explicit URL rewriting.

From the sticky information, the load balancer plug-in first determines the instance to which the request was previously forwarded. If that instance is found to be healthy, the load balancer plug-in forwards the request to that specific application server instance. Therefore, all requests for a given session are sent to the same application server instance.

The load balancer plug-in uses the following methods to determine session stickiness:

- [Cookie Based Method](#)
- [Explicit URL Rewriting Method](#)

### *Cookie Based Method*

In the cookie based method, the load balancer plug-in uses a separate cookie to record the route information.

---

<b>NOTE</b>	The HTTP client must support cookies to use the cookie based method.
-------------	--

---

### *Explicit URL Rewriting Method*

In the explicit URL rewriting method, the sticky information is appended to the URL. This method works even if the HTTP client does not support cookies.

## Configuring HTTP Load Balancing and Failover

The load balancer plug-in is an enhanced version of the HTTP reverse proxy plug-in used in Sun Java System Application Server, Standard Edition. For more information on the reverse proxy plug-in, see [Chapter 12, “Configuring the Web Server Plug-in.”](#)

This section discusses the following topics:

- [Configuration Changes to Web Server](#)
- [Creating Load Balancer Configuration File](#)
- [Configuring Multiple Web Server Instances](#)

## Configuration Changes to Web Server

The load balancer plug-in installation program makes a few modifications to the web server’s configuration files. This section details the changes to the following web server’s configuration files:

- [Modifications to Sun Java System Web Server](#)
- [Modifications to Apache Web Server](#)

---

**NOTE** The load balancer plug-in can be installed either along with Sun Java System Application Server Enterprise Edition, or separately, on a machine running the supported web server.

For complete details on the installation procedure, see *Sun Java System Application Server Installation Guide*.

---

## Modifications to Sun Java System Web Server

The installation program makes the following changes to the Sun Java System Web Server's configuration files:

1. Adds the following load balancer plug-in specific entries to the web server instance's `magnus.conf` file:

```
##EE lb-plugin
Init fn="load-modules"
shlib="webserver_install_dir/plugins/lbplugin/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrough"
Thread="no"

Init fn="init-passthrough"

##end addition for EE lb-plugin
```

2. Adds the following entries specific to the load balancer plug-in to the web server instance's `obj.conf` file:

```
<Object name=default>

NameTrans fn="name-trans-passthrough" name="lbplugin"
config-file="webserver_install_dir/webserver_instance/config/loadbalancer.xml"

<Object name="lbplugin">
ObjectType fn="force-type" type="magnus-internal/lbplugin"
PathCheck fn="deny-existence" path="*/WEB-INF/*"
Service type="magnus-internal/lbplugin" fn="service-passthrough"
Error reason="Bad Gateway" fn="send-error"
uri="$docroot/badgateway.html"
</object>
```

`lbplugin` is a name that uniquely identifies the Object, and `webserver_install_dir/webserver_instance/config/loadbalancer.xml` is the location of the XML configuration file for the virtual server on which the load balancer is configured to run.

You must now configure the load balancing plug-in, by editing the `loadbalancer.xml` configuration file. For more information on editing the `loadbalancer.xml` file, see [“Creating Load Balancer Configuration File” on page 413](#).

## Modifications to Apache Web Server

Before installing the load balancer plug-in on Apache, see information on compiling and configuring Apache in [Appendix C, “Compiling and Configuring Apache Web Server \(Enterprise Edition\)”](#).

The load balancer plug-in installation program extracts the necessary files to the `modules` folder under the web server’s root directory. It adds the following entries specific to the load balancer plug-in to the web server instance’s `httpd.conf` file:

```
<VirtualHost machine_name:443>

##Addition for EE lb-plugin

LoadFile /usr/lib/libCstd.so.1

LoadModule apachelbplugin_module libexec/mod_loadbalancer.so
#AddModule mod_apachelbplugin.cpp
<IfModule mod_apachelbplugin.cpp>
    config-file webserver_instance/conf/loadbalancer.xml
    locale en
</IfModule>

<VirtualHost machine_ip_address>
DocumentRoot "webserver_instance/htdocs"
ServerName server_name
</VirtualHost>

##END EE LB Plugin ParametersVersion 7
```

**NOTE**

- Apache web server must be run in secure mode to ensure that it works well with the load balancer plug-in.
- The URL pattern should be set to a definite context root, such as */fortune* to ensure that all requests are forwarded to the correct application. If the request cannot match any of the installed applications, then it will be redirected to the web server.
- Create a directory called *sec\_db\_files* under *apache\_install\_dir* and copy *<SIAS\_installables\_dir/sec\_db\_files/\** to *apache\_install\_dir/sec\_db\_files*.
- When more than one Apache child processes are running, each process will have its own load balancing round robin sequence.

For example, if there are two Apache child processes running, and the load balancing plug-in load balances on to two application server instances, the first request will be sent to instance #1 and the second request will also be sent to instance #1. The third request will be sent to instance #2 and the fourth request will be sent to instance #2 again. This trend will repeat (instance1, instance1, instance2, instance2, etc.)

This behavior is different from what the user would usually expect, that is, instance1, instance2, instance1, instance2 .... This behavior is due to the fact that in Sun Java System Application Server Enterprise Edition, the load balancing plug-in Apache would instantiate a load balancer instance for each apache process, thereby creating an independent load balancing sequence.

- The use of load balancer plug-in is supported for development systems only and not for production systems.

This completes the installation of load balancer plug-in on Apache Web Server.

This following section describes how to create a *loadbalancer.xml* file and specify the necessary configuration details.

## Creating Load Balancer Configuration File

The *loadbalancer.xml* file should be created based on the *sun-loadbalancer\_1\_1.dtd* and *loadbalancer.xml.example* files installed during load balancer installation. These files are available in the following location:

- On Sun Java System Web Server:  
*webserver\_install\_dir/webserver\_instance/config* directory.
- On Apache: *webserver\_install\_dir/webserver\_instance/conf* directory.

For more information on the elements and their attributes, see [Appendix E, “HTTP Load Balancer Configuration File.”](#)

*Sample loadbalancer.xml file*

```

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun Java
System Web Server 6.0//EN"
"file:///webserver_install_dir/webserver_instance/config/sun-loadbalancer_
1_1.dtd">

<loadbalancer>

<cluster name="cluster1">

<instance name="server1" enabled="true" listeners="http://machine1.sun.com:80
https://http://machine1.sun.com:443"/>

<instance name="server2" enabled="true" disable-timeout-in-minutes="31"
listeners="http://machine2.sun.com:80 https://machine2.sun.com:443"/>

<web-module context-root="webapps-simple" enabled="true"
error-url="webserver_installdir/error/error1.html">

<idempotent-url-pattern url-pattern="/simple*" no-of-retries="10"/>

</web-module>

<health-checker url="/" interval-in-seconds="60" timeout-in-seconds="10"/>

</cluster>

<property name="response-timeout-in-seconds" value="120"/>

<property name="reload-poll-interval-in-seconds" value="180"/>

<property name="https-routing" value="true"/>

<property name="require-monitor-data" value="true"/>

</loadbalancer>

```

**NOTE**

The following requirements should be considered while configuring the load balancer plug-in:

- One load balancer instance cannot have two clusters with the same name.
- One cluster cannot have two `instance` elements with the same name.
- One cluster cannot have two `web-module` elements with the same context-root.
- Application Server instance names are unique across servers/clusters.

The `listeners` attribute is a space-separated list of URLs of the application server instance's listeners. Therefore, no server instance either in a cluster, or across clusters, can have the same listener URL.

The `listeners` attribute should be in the following format:

`http://hostname.domain:port`

For example:

`http://austen.sun.com:80`

In this example, `http://austen.sun.com`, is the address of the application server instance followed by it's port number, 80.

- `instance` name, such as, `server1`, are variables and you can provide any appropriate name.
- For HTTPS routing, we recommend that you create a dedicated HTTPS listener under each application server instance.

For example,

`listeners="http://austen.sun.com:80 https://austen.sun.com:443"`

In this example, `https://austen.sun.com`, is the address of the application server instance followed by it's port number, 443.

- You must specify the exact context root in the `web-module` element. Do not use a very general context root such as `/*`.

---

Save the changes made to the `loadbalancer.xml` file and restart your web server.

**NOTE**

- For Apache Web Server, the DTD file will be in the `conf` directory. Therefore, the target location of the DTD file should be specified as given in the following example:  
`"file:///webserver_install_dir/webserver_instance/conf/sun-loadbalancer_1_1.dtd">`
- For more information on creating and configuring application server instances, see [Chapter 4, “Using Sun Java System Application Server Instances.”](#)
- The `response-timeout-in-seconds` property specifies the timeout interval within which a response must be obtained for a request that is load balanced, or an error page will be displayed.
- For information on clustering, see [Chapter 19, “Managing Clusters \(Enterprise Edition\).”](#)

For more information on `loadbalancer.xml`, see [Appendix E, “HTTP Load Balancer Configuration File.”](#)

## Configuring Multiple Web Server Instances

The Sun Java System Application Server installer will not allow the installation of multiple load balancer plug-ins on a single machine. If you need to have multiple web servers (with the load balancer plug-in), on a single machine, in either a single cluster or multiple clusters, a few manual steps are required to configure the load balancer plug-in.

Follow the procedure listed in this section to configure the load balancer plug-in for multiple web server instances.

- [Configuring Multiple Sun Java System Web Server Instances](#)
- [Configuring Multiple Apache Web Server Instances](#)



## Configuring Multiple Sun Java System Web Server Instances

1. Make the following changes to the `magnus.conf` file of the web server instance on which you want to configure the load balancer plug-in:

```
Init fn="load-modules"
shlib="webserver_install_dir/plugins/lbplugin/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrough"
Thread="no"
```

```
Init fn="init-passthrough"
```

2. Make the following changes to the `obj.conf` file of the web server instance on which you want to configure the load balancer plug-in:

```
<Object name=default>
```

```
NameTrans fn="name-trans-passthrough" name="lbplugin"
config-file="webserver_install_dir/webserver_instance/config/loadbalancer.xml"
```

```
<Object name="lbplugin">
```

```
ObjectType fn="force-type" type="magnus-internal/lbplugin"
```

```
PathCheck fn="deny-existence" path="*/WEB-INF/*"
```

```
Service type="magnus-internal/lbplugin" fn="service-passthrough"
```

```
Error reason="Bad Gateway" fn="send-error"
```

```
uri="$docroot/badgateway.html"
```

```
</object>
```

3. Copy the `sun-loadbalancer_1_1.dtd` and `loadbalancer.xml` file from the existing (which is configured with the load balancer plug-in) web server instance's `config` directory to new, load balancer configured, instance's `config` directory.
4. Edit the `loadbalancer.xml` file as described in [“Creating Load Balancer Configuration File” on page 413](#).

## Configuring Multiple Apache Web Server Instances

1. Add the following entries specific to the load balancer plug-in to the web server instance's `httpd.conf` file:

```
<VirtualHost machine_name:443>

##Addition for EE lb-plugin

LoadFile /usr/lib/libCstd.so.1

LoadModule apachelbplugin_module libexec/mod_loadbalancer.so
#AddModule mod_apachelbplugin.cpp
<IfModule mod_apachelbplugin.cpp>
config-file webserver_instance/conf/loadbalancer.xml
locale en
</IfModule>

<VirtualHost machine_ip_address>
DocumentRoot "webserver_instance/htdocs"
ServerName server_name
</VirtualHost>

##END EE LB Plugin Parameters
```

2. Copy the `sun-loadbalancer_1_1.dtd` and `loadbalancer.xml` file from the existing (which is configured with the load balancer plug-in) web server instance's `conf` directory to new, load balancer configured, instance's `conf` directory.
3. Edit the `loadbalancer.xml` file as described in [“Creating Load Balancer Configuration File” on page 413](#).

## Administering HTTP Load Balancing and Failover

After setting up HTTP load balancing as described in [Configuring HTTP Load Balancing and Failover](#), you should configure the load balancing plug-in to suit your operating environment. In most cases, changes to the load balancer plug-in can be made without stopping or restarting Application Server instances.

Therefore, you can adjust the load balancer's behavior depending on fluctuating, external factors such as hardware failures or increase in traffic volume. This section describes procedures to configure and manage the behavior of the load balancing plug-in. All changes are made in the load balancer plug-in's configuration file, `loadbalancer.xml`.

This section describes the following tasks:

- [Enabling/Disabling Application Server Instances](#)
- [Enabling Dynamic Reconfiguration](#)
- [Configuring HTTP/S Session Failover](#)
- [Configuring Response Timeout](#)
- [Configuring Quiescence](#)
- [Configuring The Health-Checker](#)
- [Configuring Idempotent URL Requests](#)
- [Configuring HTML Error Pages](#)

## Enabling/Disabling Application Server Instances

Even if you start an Application Server instance and add it to a cluster, the application server instance will not receive requests from the load balancer until you *enable* it.

- To enable an Application Server instance that has been added to the cluster, set the value of the `enabled` attribute of the `instance` element to `true` in the `loadbalancer.xml` file.
- To disable an Application Server instance which is part of a cluster, set the value of the `enabled` attribute of the `instance` element to `false` in the `loadbalancer.xml` file.

To disable an Application Server instance (or a web application) gracefully, that is, without affecting the requests it is currently processing, you must configure the quiescing time interval. For more information quiescing, see [“Configuring Quiescence” on page 423](#).

---

<b>NOTE</b>	Enabling an application server instance is different from starting the Application Server instance. Similarly, disabling an Application Server instance is different from stopping the Application Server instance.
-------------	---

---

## Restarting Application Server Instances

When you make configuration changes to an Application Server instance that requires a restart, you must follow the correct sequence of operations:

1. Disable the Application Server instance.

Set the value of the `enabled` attribute of the `instance` element to `false` in the `loadbalancer.xml` file.

2. Stop the Application Server instance.

Use either `asadmin` or the Admin Console, as described in [Chapter 1, “Getting Started with Sun Java System Application Server Administration.”](#)

3. Start the Application Server instance.

Use either `asadmin` or the Admin Console, as described in [Chapter 1, “Getting Started with Sun Java System Application Server Administration.”](#)

4. Enable the Application Server instance.

Set the value of the `enabled` attribute of the `instance` element to `true` in the `loadbalancer.xml` file.

For more information on such scenarios, see [“Reconfiguring an Application Server Instance in a Running Cluster” on page 461.](#)

---

<b>NOTE</b>	After you mark an application server instance in the <code>loadbalancer.xml</code> file as enabled or disabled, the instance is enabled or disabled when the load balancer next polls the <code>loadbalancer.xml</code> file. For more information, see <a href="#">“Enabling Dynamic Reconfiguration” on page 420.</a>
-------------	---

---

## Enabling Dynamic Reconfiguration

The load balancer plug-in detects changes to its configuration by examining the time stamp of the `loadbalancer.xml` file. If a change is detected, the load balancer automatically reconfigures itself based on the values specified in the `reload-poll-interval-in-seconds` property.

To set the duration after which time the load balancer will check for configuration changes, do the following:

1. Open the `loadbalancer.xml` file for editing.
2. Edit the value of the property `reload-poll-interval-in-seconds` property.

By default, this value is set to 0 seconds, which implies that dynamic reconfiguration is not enabled.

3. Change the value to, for example, 180 seconds.
4. Save the changes.

The load balancer will now check the `loadbalancer` file every 180 seconds for any changes.

Note that the reload poll interval has a direct bearing on the quiescing period. If the load balancer's reload interval is less than the quiescing period, the load balancer reloads the entire configuration and *starts the quiescing process again*, before quiescing can be completed.

For more information on quiescing, see [“Configuring Quiescence” on page 423](#).

---

#### NOTE

- If the changes to the `loadbalancer.xml` file are not in the correct format, as specified by the `sun-loadbalancer_1_0.dtd`, the reconfigure operation will fail. The failure notice will be printed in 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, then it uses the configuration that is currently in memory. The load balancer also ensures that the modified configuration data is compliant with the DTD before over writing the existing configuration.

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

Sun Java System Web Server's error log file can be found at this location:  
`webserver_install_dir/webserver_instance/logs/`.

---

## Configuring HTTP/S Session Failover

This section describes how to configure the load balancer plug-in to enable HTTP/HTTPS routing and session fail over between the web server and Sun Java System Application Server. The load balancer plug-in fails over HTTP /HTTPS sessions over to another application server instance if the original application server instance to which the session was connected becomes unavailable.

The following topics are discussed in this section:

- [About HTTPS Routing](#)
- [Configuring HTTPS Routing](#)

### About HTTPS Routing

All incoming requests, whether HTTP or HTTPS (secure HTTP), are routed by the load balancer plug-in to application server instances. However, if HTTPS routing is enabled, a HTTPS request will be forwarded by the load balancer plug-in to the application server using an HTTPS port only. Note that HTTPS routing is performed on both new and sticky requests.

If an HTTPS request is received and no session is in progress, then the load balancer plug-in selects an available application server instance with a configured HTTPS port, and forwards the request to that instance.

In an ongoing HTTP session, if a new HTTPS request for the same session is received, then the session and sticky information saved during the HTTP session is used to route the HTTPS request. The new HTTPS request is routed to the same server where the last HTTP request was served, but on the HTTPS port.

### Configuring HTTPS Routing

The `https-routing` property controls whether HTTPS routing is turned on or off for all the application servers that are participating in load balancing. If this property is set to `false`, then all HTTP and HTTPS requests are forwarded as HTTP.

To configure HTTPS routing and failover, perform the following steps:

- Change the value of the property `https-routing` to `true`.

For example, `<property name="https-routing" value="true"/>`.

The `https-routing` property controls whether HTTPS routing is turned on or off for all the application servers that are participating in load balancing. If this property is set to `false`, then all HTTP and HTTPS requests are forwarded as HTTP.

---

**NOTE**

- For HTTPS routing to work, there should be one or more HTTPS listeners configured.
  - If `https-routing` is set to `true`, and a new or a sticky request comes in where there are no healthy HTTPS listeners in the cluster, then that request generates an error.
- 

## Configuring Response Timeout

The `response-timeout-in-seconds` property specifies the timeout interval within which a response must be obtained for a request that is load balanced, or an error page will be displayed. The default value is 60 seconds.

While configuring `response-timeout-in-seconds`, you must take into account the maximum time-outs for all the applications that are running. The value should be set to the maximum response time of all the applications.

## Configuring Quiescence

Quiescing is the process of gracefully disabling an application server instance or web application. You can set a duration, called the quiescing period, during which time the load balancer plug-in fails over sticky requests to another healthy instance in the cluster.

If you want to stop an application server instance for any reason (for example to add a JDBC resource), you would want this instance to complete serving these requests before you stop it. Similarly, if you want to undeploy a web application from a cluster, you would want this application to complete serving the requests before undeploying it. You can achieve this by using the quiescing process.

For detailed information on quiescence, see [Chapter 19, “Managing Clusters \(Enterprise Edition\).”](#)

You can choose to quiesce an application or an application server instance. These two options are described in the following sections:

- [Quiescing an Instance](#)
- [Quiescing an Application](#)

## Quiescing an Instance

The load balancer uses the following policy for quiescing application server instances:

- If an instance is disabled, and the time out has not expired, then sticky requests will continue to be delivered to that instance. New requests, however, will not be sent to the disabled instance.
- When the time-out expires, the instance is disabled. The load balancer does not send any requests to this instance, even if all sessions sticking to this instance were not invalidated. All open connections from the load balancer to the instance are closed. Instead, the load balancer will fail over sticky requests to another healthy instance in the same cluster.

Follow these steps to quiesce an instance:

1. In the `loadbalancer.xml` file, set the `enabled` sub-element under the `instance` element to `false`.
2. Set an appropriate value for the `disable-timeout-in-minutes` sub-element.

The `disable-timeout-in-minutes` sub-element represents the elapsed time after which the load balancer will disable the specified instance. The default is 31 minutes.

## Quiescing an Application

The load balancer uses the following policy for quiescing applications:

- If an application is disabled, and the time out has not expired, new requests to the disabled applications will not be forwarded by the load balancer. These requests will be returned to the web server. Sticky requests will continue to be forwarded till the time-out expires.
- When the time out expires, the application is disabled. The load balancer will not accept any requests for this application, including sticky requests.

---

When you quiesce an application, the users of the disabled application will experience loss of service until the application is re-enabled.

---



Follow these steps to quiesce an application:

1. In the `loadbalancer.xml` file, set the `enabled` sub-element under the `web-module` element to `false`.
2. Set an appropriate value for the `disable-timeout-in-minutes` sub-element.

The `disable-timeout-in-minutes` sub-element represents the elapsed time after which the load balancer will disable the specified application. The default is 31 minutes.

## Configuring The Health-Checker

The load balancer periodically checks all the configured Sun Java System Application Server instances that are marked as unhealthy, based on the values specified in the `health-checker` element. Enabling the health checker is optional. If the health-checker is not enabled, periodic health check of unhealthy instances will not be performed.

The load balancer's health check mechanism communicates with the application server instance using HTTP. The health checker sends an HTTP request to the URL specified and waits for a response. The status code in the HTTP response header should be between 100 and 500 to consider the instance to be healthy.

To enable the health checker, edit the following properties in the `loadbalancer.xml` file:

- `url`  
Specifies the listener's URL that the load balancer will check to determine its state of health.
- `interval-in-seconds`  
Specifies the interval at which health checks of instances occur. The default is 30 seconds.
- `timeout-in-seconds`  
Specifies the timeout interval within which a response must be obtained for a listener to be considered healthy. The default is 10 seconds.

If an application server instance is marked as unhealthy, the health checker polls the unhealthy instances to determine if the instance has become healthy. The health checker uses the `url` attribute to check all unhealthy application server instances to check if they have returned to the healthy state.

If the health checker finds that an unhealthy instance has become healthy, that instance is added to the list of healthy instances.

## Configuring Idempotent URL Requests

To enhance the availability of your applications deployed on a Sun Java System Application Server cluster, configure the load balancer to retry failed idempotent HTTP requests on all the application server instances in a cluster. This option can be used for read-only requests, for example, to retry a search request.

This section describes the following topics:

- [Specifying an Idempotent URL](#)
- [Characteristics of an Idempotent URL](#)

### Specifying an Idempotent URL

To configure idempotent URL response, specify the URLs that can be safely retried in the `idempotent-url-pattern` sub-element on the `web-module` element. For example:

```
<idempotent-url-pattern url-pattern="sun_java/*"  
no-of-retries="10"/>
```

Two attributes you need to specify are `url-pattern` and `no-of-retries`.

**url-pattern** The URL pattern of the idempotent requests.

The idempotent URL can be specified as an URL pattern using wild cards. The URL pattern should conform to the mappings specified in the section, SRV 11.2, of the Servlet specification.

For example, `/foo/bar/*` (to retry URLs of the type `/foo/bar/index.html`) or `*.jpg` (to retry URLs of the type `/gallery/racecar.jpg`).

**no-of-retries** The number of retries the load balancer should perform on different Application Server instances.

The default is -1. This would ensure that in the event of a failure, all the listeners on instances that are configured to handle the application URL are tried for serving the idempotent request.

If none of the server instances can successfully serve the request, the error page as specified by the user will be returned. To configure custom error pages, see [“Configuring HTML Error Pages” on page 427](#)

## Characteristics of an Idempotent URL

Since all requests for a given session are sent to the same application server instance, and if that Application Server instance is unreachable, the load balancer will return an error message. Normally, the request will not be retried on another Application Server instance. However, if the URL pattern matches that specified in `loadbalancer.xml` file, then the request will be implicitly retried on another Application Server instance in the cluster.

An idempotent request is one that does not cause any change or inconsistency in an application, when retried. In HTTP, some methods (such as GET) are idempotent, while other methods (such as POST) are not. In effect, retrying an idempotent URL should not cause values to change on the server or in the database. The only difference should be a change in the response received by the user.

Examples of idempotent requests include, search engine queries and database queries. The underlying principle is that the retry would not cause an update or modification of data.

A search engine, for example, will send HTTP requests with the same URL pattern to the load balancer. Specifying the URL pattern of the search request to the load balancer ensures that HTTP requests with the specified URL pattern will be implicitly retried on another Application Server instance.

For example, if the request URL sent to the Application Server is of the type `/search/something.html`, then the URL pattern can be specified as `/search/*`.

Examples of non-idempotent requests include banking transactions and online shopping. If you retry such requests, chances are that money will be transferred twice from your account or you might unwittingly place a second order for, say pizzas.

## Configuring HTML Error Pages

A new sub-element, `error-url` has been added to the `web-module` element of the loadbalancer configuration file. This option allows you to specify your own error page or a URL to an error page to be displayed to the end-user.

```
<web-module context-root="webapps-simple" enabled="true"
error-url="webserver_installdir/error/error1.html">
```

To specify a custom error page, use the `error-url` sub-element. If this sub-element is specified, it overrides all other mechanisms configured for error reporting.

---

**NOTE** The URL should not point to an URL on the Application Server instance since it would not be possible to load the `error-url` if the server is down. Instead, you should specify a URL that points to a location on the Web Server.

---

Sun Java System Application Server provides the following options for specifying the error page.

- You could specify the `error-url` to be either the name of an HTML page in the standard load balancer plug-in's error pages directory, or an HTTP URL. If you specify an HTTP URL, the load balancer will forward the client request to the specified error URL.

You should not specify the location of the error page to an absolute path in the local file system. The location must be relative to the `webserver_install_dir/plugins/lbplugin/errorpages` directory.

- If you do not specify the `error-url` sub-element in the `loadbalancer.xml` file, then a default error page, `sun-http-lberror.html`, is picked up from the standard error pages directory, `errorpages`, if present.

As part of the load balancer plug-in installation, a directory called `errorpages` is created in the Web Server installation directory.

## Rules Applicable for Displaying Error Pages

The error page that is displayed on the following rules:

- When an error is encountered for an application, the load balancer will try to identify the specific `web-module` entry in `loadbalancer.xml`, based on the request URL. The load balancer will first check if the `error-url` sub-element has been defined for the web-module. If it has been defined, it will read the URL attribute and load the user-specified error page.
- If the `error-url` sub-element is missing or invalid, the load balancer will display the default error page, `sun-http-lb-error.html`, from the `errorpages` directory of the load balancer plug-in.
- If the `error-url` has been defined for a `web-module` but if that page is missing, the load balancer will load the default error page, `sun-http-lb-error.html`.
- If the default error page is missing, the error will be forwarded to the web server.

# Monitoring the Load Balancer Plug-in

This section discusses the following topics:

- [Configuring Log Messages](#)
- [Configuring Monitoring](#)
- [Understanding Monitoring Messages](#)

## Configuring Log Messages

The load balancer plug-in uses the web server's logging mechanism to write log messages. The default log level on the Application Server is set to the default logging level on Sun Java System Web Server (`INFO`), and Apache Web Server (`WARN`). The application server log levels - `FINE`, `FINER`, and `FINEST` map to the `DEBUG` level on the web server.

These log messages are written to the web server log files, and are in the form of raw data that can be parsed using scripts, or imported into spreadsheets to calculate the metrics you need.

### Types of Log Messages

The load balancer plug-in generates the following three different sets of log messages:

- [Load Balancer Configurator Log Messages](#)
- [Request Dispatch and Runtime Log Messages](#)

#### *Load Balancer Configurator Log Messages*

These messages will be logged when the load balancer configurator parses `loadbalancer.xml` for idempotent URLs and error page settings.

An output for idempotent URL pattern configuration will contain the following:

- When the log level is set to `FINE`:

```
CONFxxxx: IdempotentUrlPattern configured <url-pattern>
<no-of-retries> for web-module : <web-module> {LEVEL = FINE}
```

- When the log level is set to `SEVERE`:

```
CONFxxxx: Duplicate entry of Idempotent URL element <url-pattern>
for webModule <web-module> in loadbalancer.xml." {LEVEL = SEVERE}
```

- When the log level is set to WARN:

```
CONFxxxx: Invalid IdempotentUrlPatternData <url-pattern> for
web-module <web-module> {LEVEL = WARN}
```

An output for error page URL configuration will contain the following information:

```
CONFxxxx: Invalid error-url for web-module <web-module> {LEVEL = WARN}
```

### *Request Dispatch and Runtime Log Messages*

These log messages are generated while a request is being load balanced and dispatched.

- An output for standard logging for each method start will contain the following information:

```
ROUTxxxx: Executing Router method <method_name> {LEVEL = FINE}
```

- An output for router logs for each method start will contain the following information:

```
ROUTxxxx: Successfully Selected another ServerInstance for
idempotent request <Request-URL> {LEVEL = INFO}
```

- An output for runtime logs will contain the following information:

```
RNTMxxxx: Retrying Idempotent <GET/POST/HEAD> Request <Request-URL>
{LEVEL = INFO}
```

### Configurator Error Messages

- ROUTxxxx: Non Idempotent Request <Request-URL> cannot be retried {LEVEL = INFO}

```
Example :: ROUTxxxx: Non Idempotent Request
http://abc.com/addToDB?x=11&abc=2 cannot be retried
```

- RNTMxxxx: Invalid / Missing Custom error-url / page: <error-url> for web-module: <web-module> {LEVEL = FINE}

```
Example :: RNTMxxxx: Invalid / Missing Custom error-url / page:
myerror1xyz for web-module: test
```

## Configuring Monitoring

Perform the following steps to turn on load balancer plug-in log messages:

1. From Sun ONE Web Server's admin console, go to the `Magnus Editor` tab.
2. Set the `Log Verbose` option to `On`.
3. Set the value of the `require-monitor-data` in `loadbalancer.xml` property to `true`.

For example: `<property name="require-monitor-data" value="true" />`

The load balancer plug-in logs record the following information:

4. Request start/stop information is logged for every request.
5. Failed over request information is logged when the request fails over from an unhealthy instance to a healthy instance.
6. List of unhealthy instances is logged at the end of every health check cycle.

---

### NOTE

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`.

---

## Understanding Monitoring Messages

The format of the load balancer plug-in log messages is as follows.

- The start of an HTTP request contains the following information:

```
RequestStart Sticky(New) <req-id> <time-stamp> <URL>
```

The timestamp value is the number of milliseconds from January 1, 1970. For example:

```
RequestStart New 123456 602983
http://austen.sun.com/Webapps-simple/servlet/Example1
```

- The end of an HTTP request contains the `RequestExit` message, as follows:

```
RequestExit Sticky(New) <req-id> <time-stamp> <URL> <listener-id>
<response-time> Failure-<reason for error>(incase of a failure)
```

For example:

```
RequestExit New 123456 603001
http://austen.sun.com/Webapps-simple/servlet/Example1
http://austen:2222 18
```

---

**NOTE**

In the `RequestExit` message, `<response-time>` indicates the total request turn-around time in milliseconds, from the perspective of the load balancer plug-in.

---

- The list of unhealthy instances, as follows:

```
UnhealthyInstances <cluster-id> <time-stamp> <listener-id>,
<listener-id>...
```

For example:

```
UnhealthyInstances cluster1 701923 http://austen:2210, http://austen:3010
```

- A list of failed-over requests, as follows:

```
FailedoverRequest <req-id> <time-stamp> <URL> <session-id>
<failed-over-listener-id> <unhealthy-listener-id>
```

For example:

```
FailedoverRequest 239496 705623
http://austen.sun.com/Apps/servlet/SessionTest 16dfdac3c7e80a40
http://austen:4044 http://austen:4045
```

## Known Issues in Load Balancing Requests

The following list discusses the limitations in Load Balancer with respect to HTTP/HTTPS request processing.

- If a session uses a combination of HTTP and HTTPS requests, then the first request has to be a HTTP Request. If the first request is an HTTPS request, it cannot be followed by an HTTP request. This is because the cookie associated with the HTTPS session is not returned by the browser. The browser interprets the two different protocols as two different servers, and initiates a new session.



This limitation is valid only if `https-routing` is set to `true`.

- If a session has a combination of HTTP and HTTPS requests, then the application server instance has to be configured with both HTTP and HTTPS listeners.

This limitation is valid only if `https-routing` is set to `true`.

- If a session has a combination of HTTP and HTTPS requests, then the application server instance has to be configured with HTTP and HTTPS listeners that use standard port numbers, that is, 80 for HTTP, and 443 for HTTPS.

This limitation is valid regardless of the value set for `https-routing`.



# Configuring RMI/IIOP Load Balancing and Failover (Enterprise Edition)

This chapter describes how to setup load balancing and failover of RMI-IIOP requests.

This chapter includes the following topics:

- [About RMI/IIOP Load Balancing and Failover](#)
- [Configuring RMI/IIOP Failover](#)
- [Sample server.xml Configuration](#)

## About RMI/IIOP Load Balancing and Failover

Sun Java™ System Application Server provides high availability of remote EJB references and name service objects over RMI-IIOP.

This section discusses the following topics:

- [Requirements for Load Balancing and Failover](#)
- [About RMI/IIOP Load Balancing and Failover Algorithm](#)

## Requirements for Load Balancing and Failover

The following section lists the requirements for setting up load balancing and failover over RMI/IIOP:

- Sun Java System Application Server Enterprise Edition 7 is installed.
- At least two application server instances are created and RMI/IIOP endpoints configured in a cluster. For more information on configuring IIOP clusters, see [“Configuring RMI/IIOP Failover” on page 437](#).
- Deployed your J2EE applications to all application server instances that will participate in load balancing.
- Enabled your RMI/IIOP applications for load balancing.

For more information on configuring RMI/IIOP client applications for load balancing, see *Sun Java System Application Server Developer's Guide to Clients*.

---

**NOTE**

- RMI/IIOP endpoints and HTTP listeners (if you are using HTTP load balancing) must be configured on the same cluster of Application Server instances.
  - For more information on clusters, see [Chapter 19, “Managing Clusters \(Enterprise Edition\)”](#).
  - For more information on HTTP Load Balancing, see [Chapter 17, “Configuring HTTP Load Balancing and Failover \(Enterprise Edition\)”](#).
- 

## About RMI/IIOP Load Balancing and Failover Algorithm

High availability is achieved by enabling RMI/IIOP clients for load balancing and configuring the Application Server for failover of remote references.

The configuration settings required to enable RMI/IIOP based applications depends on the type of client. For detailed information on setting up various RMI/IIOP clients for load balancing, see *Sun Java System Application Server 7 Developer's Guide to Clients*.

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.

For detailed information on setting up RMI/IIOP clients for load balancing, see *Sun Java System Application Server Developer's Guide to Clients*.

---

**NOTE** Failover of remote references for servlets and JSPs in web applications executing in a JVM different from the target Application Server instance, is not supported.

---

## Initial Context Based Failover Algorithm

Sun Java System Application Server Enterprise Edition 7 employs `InitialContext` based algorithm for load balancing and failover of remote references of EJBs and name service objects on the RMI/IIOP path.

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. One of the available IIOP endpoints (on the server instance selected in the IIOP cluster) is designated as the *primary endpoint*. The other IIOP endpoints in the cluster are designated as *alternate endpoints*.

The name service object used to create the `InitialContext` is then used to service all subsequent requests with the same `InitialContext`. Subsequent operations with this `InitialContext` will be *assigned* or *stuck* to the same Application Server instance.

If the primary endpoint corresponding to the remote reference becomes unavailable, further operations corresponding to this remote reference is failed over to one of the alternate endpoints.

---

**NOTE** If an Application Server instance becomes unavailable while processing RMI/IIOP requests, that request will not be retried or failed over to another Application Server instance.

---

## Configuring RMI/IIOP Failover

To configure high availability over RMI/IIOP, each Application Server instance needs to be configured with the available IIOP listeners that's used for serving RMI-IIOP requests. The IIOP cluster configuration information is registered in `server.xml`.

For detailed information on configuring RMI/IIOP clients for load balancing, see *Sun Java System Application Server 7 Developer's Guide to Clients*.

---

**NOTE** Only IIOP listeners, configured under the ORB tab of the Admin Console, can be configured as IIOP endpoints.

---

Setting up a cluster and assigning endpoints can be performed from both the administration interface and command line. The user, via the Administration CLI or the GUI, changes the `server.xml` to define IIOP endpoints that will constitute the cluster of Application Server instances.

For details on using the administration console and the command line tool, `asadmin`, see the help associated with those tools.

The following topics are discussed in this section:

- [Using Admin Console to Configure RMI/IIOP Load Balancing](#)
- [Using CLI to Configure RMI/IIOP Load Balancing](#)
- [Sample `server.xml` Configuration](#)

## Using Admin Console to Configure RMI/IIOP Load Balancing

1. In the left pane of the Administration Console, select Availability Service.
2. Expand the Availability Service node.

The right pane displays options to create and add Application Server instances to a IIOP cluster.

3. Click `New`.

The Administration console displays a page where you can specify the properties for a new IIOP Server instance (which is in-effect, the Application Server instance). Each IIOP server instance can have multiple IIOP endpoints.

The default IIOP Server Instance name is the same as the Application Server instance's name.

- a. Enter the IIOP Server Instance Name.

This is the name of the Application Server instance that will be part of the IIOP cluster. For example, `server2`. In `server.xml` the IIOP Server Instance Name is identified by the property `iiop-server-instance`.

- b. Enter the ID of the IIOP Endpoint at which the ORB listens for IIOP connections.

The IIOP Endpoint is an IIOP Listener that will be part of the IIOP Cluster. For example, orb-listener-2.

---

**NOTE** The IIOP Endpoint specified must also exist as an IIOP Listener for the ORB. If you add an endpoint on this page, you must add it as a listener for the ORB, if it has not been added already.

---

In `server.xml`, the IIOP endpoint that will be part of the IIOP Cluster is identified by the property `iiop-endpoint-id`.

- c. Enter the host ID of the Application Server instance.

The host ID is the name of the machine on which the Application Server is installed, and which hosts the `iiop-endpoint`. For example, `machine1.sun.com`.

In `server.xml`, the IIOP Endpoint is identified by the property `iiop-endpoint-host`.

- d. Enter the port number of the `iiop-endpoint`.

This is a valid ORB listener port for the IIOP Endpoint. For example, 3700. This is an optional field. If it is left blank, the Application Server will assign a valid port number.

In `server.xml`, the IIOP Endpoint port number is identified by the property `iiop-endpoint-port`.

4. Click OK to add this IIOP Server Instance to the IIOP cluster.
5. Select Apply Changes and restart the Application Server instance for the changes to take effect.

Using the same procedure add more IIOP server instances to the IIOP cluster, depending on the requirements of your applications.

## Using CLI to Configure RMI/IIOP Load Balancing

Use the following commands to set up IIOP Endpoints and Clusters from the command line:

- `clsetup`
- `asadmin deploy`
- `asadmin get`
- `asadmin set`
- `asadmin add-iiop-cluster-endpoint`
- `asadmin delete-iiop-cluster-endpoint`
- `asadmin list-iiop-cluster-config`

For detailed information on these commands and how to use them, consult the [online help](#).

## Sample server.xml Configuration

The `sun-server_1_2.dtd` has been modified to include an `iiop-cluster` element under the `availability-service` element. The `iiop-cluster` element defines the IIOP Endpoints of Application Server instances forming the IIOP cluster. When you configure an IIOP Cluster, either through the Admin Console or the CLI, the changes are registered in the `iiop-cluster` element in `server.xml`.

### Sample availability-service Configuration in server.xml

```
<availability-service availability-enabled="true">
  <iiop-cluster>
    <iiop-server-instance name="server1">
      <iiop-endpoint id="orb-listener-1" host="machine1.sun.com" port="3700"/>
      <iiop-endpoint id="orb-listener-2" host="machine2.sun.com" port="3800"/>
    </iiop-server-instance>
    <iiop-server-instance name="server2">
      <iiop-endpoint id="orb-listener-3" host="machine3.sun.com" port="3900"/>
      <iiop-endpoint id="orb-listener-4" host="machine4.sun.com" port="4000"/>
    </iiop-server-instance>
  </iiop-cluster>
</availability-service>
```



```
</iiop-server-instance>  
</iiop-cluster>  
</availability-service>
```

Sample server.xml Configuration

# Managing Clusters (Enterprise Edition)

The use of Sun Java™ System Application Server clusters provides high availability for your J2EE applications. This chapter explains how to configure and manage a group of load balanced clusters. It also provides important information about maintaining and operating clusters to fully utilize the failover capabilities of Sun Java™ System Application Server Enterprise Edition.

This chapter mainly discusses HTTP clusters. For information on working with RMI/IIOP clusters, see [“Configuring RMI/IIOP Failover” on page 437](#).

This chapter includes the following sections:

- [About Application Server Clusters](#)
- [Administering Application Server Clusters](#)
- [Online Reconfiguration of HTTP Clusters](#)
- [Using Multiple Load Balancers](#)

## About Application Server Clusters

A cluster is a group of Application Server instances that work together as one logical entity. A cluster provides a runtime environment for one or more J2EE applications.

The use of clusters in Sun Java System Application Server helps you to achieve:

- High availability by allowing for failover protection of Application Server instances in a cluster. If one Application Server instance goes down, other Application Server instances take over the requests that the unavailable server was serving.

- Scalability by allowing for the addition of Application Server instances to a cluster, thus increasing the capacity of the system. The load balancer of Sun Java System Application Server distributes requests to the available Application Server instances within the cluster.

---

**NOTE** You can use multiple clusters to perform online upgrades without loss of service on the HTTP path. This is explained in [“Using Multiple Clusters for Online Upgrades Without Loss of Service”](#) on page 460.

---

A few important facts about Sun Java System Application Server clusters are listed here:

- Sun Java System Application Server clusters support failover of HTTP and RMI/IIOP sessions. Failover of certain J2EE object references that are stored within HTTP sessions is also supported.

For more information on failover of HTTP sessions, see [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\).”](#)

- The Application Server instances in a Sun Java System Application Server cluster must be *homogenous*. That is:
  - Session persistence configuration must be identical for a given application across all instances in a cluster. Do not deploy an application with different settings for different Application Server instances in a cluster.

For these applications to take part in failover, they must be *distributable*, that is, enabled for session persistence.

For more information, see [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\).”](#)

- All the instances should point to the same session store. For more information, see [“Managing the Session Store for Clusters”](#) on page 472.
  - The value of the `cluster-id` property should be the same for all Application Server instances. For more information, see [“Managing the Session Store for Clusters”](#) on page 472.
- Each Application Server instance in a cluster can host more than one application.

The applications can serve HTTP and RMI/IIOP clients.

- Although you can add the same Application Server instance to multiple clusters, it is not recommended as management of such clusters will become very complex. For example, when an instance, which is part of more than one cluster, is removed from just one of the clusters.
- It is not mandatory that an Application Server instance be part of a cluster. However, an instance that is not part of a cluster cannot take advantage of high availability through transfer of session state from one instance to other instances.
- 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 a single cluster.
- A particular load balancer can forward requests to Application Server instances on multiple clusters. You can use this ability of the HTTP load balancer plug-in to perform an online upgrade without loss of service.

For more information, see [“Using Multiple Clusters for Online Upgrades Without Loss of Service” on page 460](#). Dynamic reconfiguration of the RMI/IIOP load balancer is not supported.

- A single cluster can receive requests from multiple load balancers. If a cluster is served by more than one load balancer, you must configure the cluster in exactly the same way on each load balancer. This means that the entries corresponding to the cluster in each `loadbalancer.xml` (for HTTP sessions) and `server.xml` (for RMI/IIOP sessions) must be identical.
- Each session is tied to a particular cluster. Therefore, though an application can be deployed on multiple clusters, sessions will only be failed over to Application Server instances within the same cluster.

The cluster thus acts as a safe boundary for session failover for the Application Server instances within the cluster.

## Defining a Cluster

To define a cluster, specify a name for the cluster in the `loadbalancer.xml` file. You also set up other parameters for the cluster.

Use the `name` attribute of the `cluster` element to define the name of the cluster (cluster is a subelement of the element `loadbalancer` in the `loadbalancer.xml` file).

To define a new cluster, do the following:

1. In the `loadbalancer.xml` file, create a new `cluster` sub-element for the `loadbalancer` element.
2. Give a unique name to the `name` attribute of `cluster`.

---

**NOTE** You can create more than one cluster in the `loadbalancer.xml` file.

---

3. Specify a URL for the health checker for the cluster.

The health checker URL is suffixed to the URL of the listener for the Application Server instance to formulate the URL at which health check requests are sent to the instance. For example, if the listener URL of an instance is `http://www.example.com:80` and the health checker URL is `/fortune`, the health check requests for that instance are sent to

`http://www.example.com:80/fortune`

The health check requests are HTTP requests. The response from the instance should be in the range of 100 to 500.

You specify the healthcheck URL by using the `url` attribute of the `health-checker` element in the `loadbalancer.xml` file (the `health-checker` element is a subelement of the `cluster` element.)

4. Specify the time interval in seconds after which this health check is performed by using the `interval-in-seconds` attribute of the `health-checker` element. This is a numeric value. The default value is 30 seconds.
5. Specify the timeout value in seconds for the health check request by using the `timeout-in-seconds` attribute of the `health-checker` element. This is the time period within which the response should be returned by the Application Server instance to the health check request sent by the load balancer failing which the Application Server instance is considered unavailable. The default value is 10 seconds.

#### Code Example 19-1 Defining a Cluster

```
<cluster name=mycluster1>
<health-checker url="/" interval-in-seconds="10"
time-out-in-seconds="20"/>
</cluster>
```

## Requirements for Adding an Application Server Instance to a Cluster

The following conditions must be true before you add an Application Server instance to a cluster:

- Before you can add any Application Server instance to a cluster, you must have created the instance through the Administration interface or the command-line interface. For information on creating an Application Server instance, see [“Adding an Application Server Instance” on page 82](#).
- Any new Application Server instance that you add to a cluster must have the same version and the same patch level of Sun Java System Administration Server as all the existing Application Server instances in the cluster.
- You must ensure the availability of all required peripheral products and libraries (for example JDBC driver classes) in the correct locations and provide the needed access rights.
- The Application Server instances in a Sun Java System Application Server cluster must be *homogenous*. That is:
  - Session persistence configuration must be identical for a given application across all instances of a cluster. Do not deploy an application with one set of settings to one segment of the instances in a cluster and another set of settings to another segment of instances in the cluster.
  - All the instances should point to the same session store. For more information, see [“Managing the Session Store for Clusters” on page 472](#).
  - The value of the `cluster-id` property should be the same for all Application Server instances. For more information, see [“Managing the Session Store for Clusters” on page 472](#).
- The system time of all the servers on which Application Server instances in a cluster are deployed must be synchronized.

## Adding Application Server Instances to a Cluster

For adding Application Server instances to a cluster, use the `instance` sub element of the `cluster` element in the `loadbalancer.xml` file.

For detailed information of the attributes of the sub-element, `instance`, see [“Elements in the loadbalancer.xml File” on page 639](#).

To add an Application Server instance to a cluster, do the following:

1. In the `loadbalancer.xml` file, create a new `instance` sub-element for the cluster element.
2. Provide a name to the Application Server instance using the `name` attribute of `instance`. This name need not be the same as the name of the Application Server instance that you specified when creating the instance. This name must be unique across all the clusters referenced in a single `loadbalancer.xml` file. That is, no two Application Server instances within a cluster can have the same name.
3. Specify whether the Application Server instance is enabled or disabled by specifying `true` or `false` for the `enabled` attribute of `instance`. The default value is `true`, that is the Application Server instance is enabled.
4. Specify the quiescing period in minutes by using the `disable-timeout-in-minutes` attribute of `instance`. This is a numeric value. The default value is 31, which is one minute more than the default session idle timeout of 30 minutes.

If you need to disable an Application Server instance, this is the time interval after which the load balancer stops sending assigned requests to that instance. For more information on quiescing and quiescing period, see [“Disabling and Quiescing an Application Server Instance in a Cluster” on page 456](#).

5. Specify the listeners for the Application Server instance by providing the URL of the HTTP listener for the Application Server instance.

You provide this information using the `listeners` attribute of the `instance` element in the `loadbalancer.xml` file. You can assign one or more listeners to the Application Server instance. To specify multiple listeners for an Application Server instance, separate the URLs of the listeners with a space. For example, `https://www.example.com:41 https://www.example.com:42`.

---

**NOTE** After you add an HTTP listener to an Application Server instance, you must restart the SNMP monitoring subagent. If you do not do this, it is possible that information about the new listener will not be available through the SNMP monitoring subagent.

---

Here is an example where an Application Server instance named `myinstance1` has been added to the cluster named `mycluster1`. Two listeners have been specified for the Application Server instance. The health checker for this cluster has also been configured.



```

<cluster name="mycluster1">

<instance name="myinstance1" enabled="true" disable-timeout-in-minutes="36"
listeners="http://www.example.com:41 https://www.example.com:42">

</instance>

<health-checker url="/" interval-in-seconds="10" time-out-in-seconds="15"
/>

</cluster>

```

## Defining Multiple Clusters

You can set up a single load balancer to serve multiple clusters. To do this, set up each cluster as explained in [“Defining a Cluster” on page 445](#) and [“Adding Application Server Instances to a Cluster” on page 447](#).

---

**NOTE** When you configure the session persistence settings, create a different session store for each cluster. For more information on creating a session store for a cluster, see [“Managing the Session Store for Clusters” on page 472](#).

---

# Administering Application Server Clusters

This section describes the following topics:

- [Starting a Cluster](#)
- [Deploying a Web Application](#)
- [Undeploying a Web Application](#)
- [Redeploying an Existing Web Application on a Cluster](#)
- [Stopping an Application Server Instance in a Cluster](#)
- [Removing an Application Server Instance from a Cluster](#)
- [Stopping a Cluster](#)
- [Removing a Cluster](#)

## Starting a Cluster

To start a cluster, do the following:

1. Start all Application Server instances in the cluster.

To start all instances in the cluster, use the `cladmin` command as follows:

```
./cladmin [--instancefile instance_file_location] [--passwordfile  
password_file_location]start-instance
```

where:

*instance\_file\_location* is the location of the `clinstance.conf` file.

*password\_file\_location* is the location of the `clpassword.conf` file.

For more information on the `cladmin` command, see [Appendix F, “Using the cladmin Command for Administration \(Enterprise Edition\).”](#)

2. Enable all the Application Server instances in the cluster.

Set the value of the `enabled` attribute of the `instance` element to `true` in the `loadbalancer.xml` file.

## Deploying a Web Application

To deploy a web application on a cluster, do the following:

1. Deploy the web application on all Application Server instances in the cluster.

To deploy the web application to all instances in the cluster, use the `cladmin` command as follows:

```
./cladmin deploy [--instancefile instance_file_location] [--passwordfile  
password_file_location][--secure | -s] [--virtualservers virtual_servers]  
[--type application|ejb|web|connector] [--contextroot contextroot]  
[--force=true|false] [--precompilejsp=true|false] [--name component_name]  
[--upload=true|false] [--retrieve local_dirpath] [--instance instance_name]  
filepath
```

where:

*instance\_file\_location* is the location of the `clinstance.conf` file.

*password\_file\_location* is the location of the `clpassword.conf` file.

For more information on the `cladmin` command, see [Appendix F, “Using the cladmin Command for Administration \(Enterprise Edition\).”](#) For more information on deploying an application to an Application Server instance, see [“Tools for Deployment” on page 345.](#)

2. In the `loadbalancer.xml` file, create a new `web-module` sub-element (within the `cluster` element) for the cluster to which you want to deploy the application.
3. Provide the context root URL for the web module that you want to deploy to the cluster by using the `context-root` attribute of the `web-module` element.

---

**NOTE** One cluster cannot have more than one `web-module` element with the same `context-root`.

---

4. Specify whether the web application is enabled or disabled by specifying `true` or `false` for the `enabled` attribute of `web-module`. The default value is `true`, that is, the web application is enabled.

Typically, when you deploy applications to a cluster, you enable the application. Enabling an application in a cluster means that the load balancer will route requests to this application.

5. Specify the quiescing period in minutes by specifying a value for the `disable-timeout-in-minutes` attribute of `web-module`. This is a numeric value. The default value is 31, which is one minute more than the default session idle timeout of 30 minutes.

If you disable the web application, this is the time interval after which the load balancer stops sending assigned requests to the web application. For more information on quiescing, see [“Disabling and Quiescing a Web Application in a Cluster” on page 458.](#)

Here is an example in which a web application with the context root `/fortune` has been added to a cluster called `cluster1`.

```
<cluster name="cluster1">
  <web-module context-root="/fortune" enabled="true" />
</cluster>
```

## Undeploying a Web Application

To undeploy a web application from a cluster, do the following:

1. Disable the web application from the cluster by setting the value (in the `loadbalancer.xml` file) of the `enabled` attribute of the `web-module` element to `false`.
2. After the quiescing period is over, undeploy the web application from all the Application Server instances in the cluster.

To undeploy the web application from all instances in the cluster, use the `cladmin` command as follows:

```
./cladmin undeploy [--instancefile instance_file_location]  
[--passwordfile password_file_location][--secure | -s] [--type  
application|ejb|web|connector] [--instance instance_name]  
component_name
```

where:

*instance\_file\_location* is the location of the `clinstance.conf` file

*password\_file\_location* is the location of the `clpassword.conf` file

3. In the `loadbalancer.xml` file, remove the `web-module` sub-element (within the `cluster` element) that references the undeployed application.

---

**NOTE** After an application is undeployed, the session state information for that application is not immediately removed from the HADB. This session state information is removed by the Sun Java System Application Server in the subsequent cycle when the timed-out sessions are removed.

---

## Redeploying an Existing Web Application on a Cluster

There may be cases when you want to make changes to a deployed web application that require you to undeploy the web application and then again deploy it. For example, you may want to deploy an upgraded version of a web application. In such cases, follow these steps:

1. Disable the web application by setting the value (in the `loadbalancer.xml` file) of the `enabled` attribute of the corresponding `web-module` element to `false`.

2. After the quiescing period is over, redeploy the web application on the Application Server instances in the cluster.
3. Enable the web application in the cluster by setting the value (in the `loadbalancer.xml` file) of the `enabled` attribute of the corresponding `web-module` element to `true`.

## Stopping an Application Server Instance in a Cluster

To stop an Application Server instance in a cluster, do the following:

1. Disable the Application Server instance by setting the value (in the `loadbalancer.xml` file) of the `enabled` attribute of the `instance` element to `false`.
2. After the quiescing period is over, plus a sufficient time to allow the requests being served by the Application Server instance to complete, stop the Application Server instance.

## Removing an Application Server Instance from a Cluster

To remove an Application Server instance from a cluster, do the following:

1. Disable the Application Server instance by setting the value (in the `loadbalancer.xml` file) of the `enabled` attribute of the corresponding `instance` element to `false`.
2. After the quiescing period plus a sufficient time to allow the requests being served by the Application Server instance to complete, remove the entries corresponding to the Application Server instance in `loadbalancer.xml` file.

Once you remove an Application Server instance from a cluster, any assigned request for that instance is routed by the load balancer to other instances as part of the round-robin algorithm. If the Application Server instance to which the request is routed is part of the same cluster and the session has not been invalidated, the session is failed over. Otherwise, the Application Server instance to which the request is routed treats the request as an unassigned request.

## Stopping a Cluster

To stop a cluster, do the following:

1. Disable all Application Server instances from the cluster by setting the value (in the `loadbalancer.xml` file) of the `enabled` attribute of the corresponding `instance` element to `false`.
2. After the quiescing period is over, plus a sufficient time to allow the requests being served by the Application Server instance to complete, stop all Application Server instances that belong to the cluster.

To stop all instances in the cluster, use the `cladmin` command as follows:

```
./cladmin [--instancefile instance_file_location] [--passwordfile  
password_file_location] stop-instance
```

where:

- *instance\_file\_location* is the location of the `clinstance.conf` file
- *password\_file\_location* is the location of the `clpassword.conf` file

## Removing a Cluster

To remove a cluster from a loadbalancer, do the following:

1. Disable each web application deployed to the Application Server instances in the cluster by setting the value (in the `loadbalancer.xml` file) of the `enabled` attribute of the corresponding `web-module` element to `false`.
2. (Optional) After all the web applications have been quiesced, undeploy the web applications from all the individual Application Server instances in the cluster. To undeploy each application from all the instances in the cluster, use the `cladmin` command. For information on how to do this, see [“Using Multiple Load Balancers” on page 461](#).
3. Remove all entries for the cluster (including entries for the Application Server instances and the web applications in the cluster) from the `loadbalancer.xml` file.

## Sample Cluster Configuration in a loadbalancer.xml File

Here are sample entries for a cluster in the `loadbalancer.xml` file. Note that in a complete `loadbalancer.xml` file, there will be other entries, for example related to load balancer polling interval, HTTPS routing enabling, and so on. For more information about the other possible entries, see [“Known Issues in Load Balancing Requests” on page 432](#).

```
<loadbalancer name="loadbalancer1">
    <cluster name="cluster1">
        <instance name="myinstance1" enabled="true"
listeners="http://www.example.com:41 https://www.example.com:42">
            </instance>
        <instance name="myinstance2" enabled="true"
listeners="http://www.example.com:43 https://www.example.com:44">
            </instance>
            <web-module context-root="/fortune" enabled="true" />
            <web-module context-root="/shopping" enabled="true" />
            <health-checker url="/" interval-in-seconds="10" />
        </cluster>
    </loadbalancer>
```

## Online Reconfiguration of HTTP Clusters

Suppose an Application Server instance in a cluster has received requests and is in the process of serving them. If you now want to stop this instance for any reason (for example to add a JDBC resource), you would want this instance to complete serving these requests before you stop it. Similarly, if you want to undeploy a web application from a cluster, you would want all instances in the cluster that are serving requests for these application to complete serving the requests before you undeploy the web application. Quiescing is a process that helps achieve this.

This topic contains the following sections:

- [Disabling and Quiescing an Application Server Instance in a Cluster](#)
- [Disabling and Quiescing a Web Application in a Cluster](#)

- [Modifying the Quiescing Period While Quiescing is On](#)

## Disabling and Quiescing an Application Server Instance in a Cluster

Quiescing an Application Server instance is the process of shutting it down in a phased manner. First, the load balancer stops sending any unassigned requests to the instance and instead diverts these unassigned requests to other available instances in the cluster.

However, for a time interval called the quiescing period, the load balancer still sends to the Application Server instance the assigned requests that it was serving. For information about specifying the quiescing period for an Application Server instance, see [“Adding Application Server Instances to a Cluster” on page 447](#).

Next, after the quiescing period is over, the load balancer stops sending to the Application Server instance the assigned requests that the instance was serving and fails over these assigned requests to other available instances in the cluster.

---

<b>NOTE</b>	These assigned requests are honored only if high availability has been enabled for the corresponding web applications. For more information on enabling high availability for web applications, see <a href="#">“Making a Web Application Distributable” on page 481</a> .
-------------	--

---

Even after the quiescing period is over, the load balancer allows the instance to serve all the requests that the load balancer dispatched to the instance before the quiescing period ended. For such requests, the load balancer returns the results to the client even after the quiescing period is over. Therefore, you should wait for the quiescing period plus a sufficient time to allow the requests being served by the Application Server instance to complete before you shut down the Application Server instance.

In some cases, however, these requests may be very long running, and it is possible that some such requests will not be served when you shut down the Application Server instance.

Quiescing does not start as soon as you make changes to the `loadbalancer.xml` file and save the changes. The load balancer polls the `loadbalancer.xml` file periodically to check if that the time stamp of the file has changed from the last time polling happened. For more information, see [“Monitoring the Load Balancer Plug-in” on page 429](#). If the load balancer detects that the time stamp of the file has changed



from the last time, it reloads the entire configuration of the `loadbalancer.xml` file. Therefore, if you disable an Application Server instance, the load balancer will start quiescing the Application Server instance when it *next* polls the `loadbalancer.xml` file. Similarly, if you mark an instance as enabled in the `load balancer.xml` file, the instance will be enabled when the load balancer next polls the `loadbalancer.xml` file.

As an example, let us take an Application Server instance whose quiescing period is 31 minutes. At the beginning of the quiescing period, the load balancer stops sending all unassigned requests to the Application Server instance but continues to send the assigned requests to the Application Server instance. After 31 minutes, the loadbalancer stops sending even the assigned requests to the Application Server instance and fails over the assigned requests to other available Application Server instances in the cluster. However, if at 30 minutes and 50 seconds the load balancer had sent an assigned request to the Application Server instance, the Application Server instance will be allowed to serve this assigned request even after 31 minutes are over.

To enable quiescing, you use the `enabled` attribute of the `instance` element in the `loadbalancer.xml` file. In a running cluster, setting the value of `enabled` for an Application Server instance in that cluster to `false` in the corresponding `loadbalancer.xml` file(s) initiates quiescing for that Application Server instance (when the load balancer next polls the `loadbalancer.xml` file).

You specify the quiescing period by specifying an appropriate value for the `disable-timeout-in-minutes` subelement of the `instance` element in the `loadbalancer.xml` file. The default value of `disable-timeout-in-minutes` is 31 minutes, which is one minute more than the default `session-idle-timeout` value of 30 minutes.

You can use the load balancer logs to ascertain if the Application Server instance has been quiesced. Once an instance has been quiesced, the log will contain the following entry:

Instance: *instance\_name* quiesced successfully over the cluster: *cluster\_name*.

*instance\_name* is the name specified for the instance in the `loadbalancer.xml` file. *cluster\_name* is the name of the cluster, in which the instance exists, as specified in the `loadbalancer.xml` file.

## Disabling and Quiescing a Web Application in a Cluster

If you disable a web application on a cluster, the load balancer stops sending any unassigned request for the web application to Application Server instances in the cluster. If this web application is deployed on other clusters that the load balancer is serving, these unassigned requests are sent to Application Server instances in the other clusters.

However, assigned requests for this web application that are being serviced by Application Server instances in the cluster will continue to be serviced till the quiescing period ends. After the quiescing period is over, the load balancer stops sending even these assigned requests to Application Server instances in the cluster.

You use the `enabled` attribute of the `web-module` element in the `loadbalancer.xml` file to disable a web application in a cluster. In a running cluster, setting the value of `enabled` for a web application in that cluster to `false` disables the web application on the cluster. As in the case of Application Server instances, the quiescing does not start immediately after you make the changes and save them. The quiescing starts after the next load balancer polling occurs.

Specify the quiescing period by specifying an appropriate value for the `disable-timeout-in-minutes` subelement of the `web-module` element in the `loadbalancer.xml` file. The default value of `disable-timeout` is 31 minutes, which is one minute more than the default `session-idle-timeout` value of 30 minutes.

You can use the load balancer logs to ascertain if the web application has been quiesced. Once a web application has been quiesced, the log will contain the following entry:

Application: *application\_name* quiesced successfully over the cluster *cluster\_name* where,

- *application\_name* is the context root of the application as specified in the `loadbalancer.xml` file.
- *cluster\_name* is the name of the cluster, on which the application has been deployed, as specified in the `loadbalancer.xml` file.

## Modifying the Quiescing Period While Quiescing is On

Suppose an Application Server instance or a web application is being quiesced and that the quiescing period is more than the load balancer polling interval.

You now make some other changes to the `loadbalancer.xml` file before the next load balancer polling interval. When the load balancer next polls the `loadbalancer.xml` file, it detects a change in the time stamp from the last time it polled the file. Because the quiescing period is more than the load balancer polling period, the quiescing is not over yet. However, the load balancer reloads the entire configuration and *starts the quiescing process again*.

You can use this feature to alter the quiescing period of an Application Server instance or a web application that is being quiesced. As an example, consider the following scenario.

The quiescing period of an Application Server instance is 30 minutes and the load balancer polling interval is 1 second. You now disable the Application Server instance. When the load balancer polling occurs again, the load balancer starts to quiesce the instance.

Suppose after 10 minutes, you want to reduce the overall quiescing period from 30 minutes to 15 minutes, that is, you want the quiescing period to be over in the next 5 minutes. For this, you change the quiescing period of the Application Server instance to 5 minutes. When the load balancer polling occurs again, the load balancer quiesces the instance once more (because the quiescing is not over yet), with a new quiescing period of 5 minutes.

The overall quiescing period is therefore 15 minutes (10 minutes during the first cycle and 5 minutes during the second cycle).

Obviously, while using this feature, you cannot reduce the quiescing period to a value less than the load balancer polling interval.

For information about setting the load balancer polling interval, see [“Monitoring the Load Balancer Plug-in” on page 429](#).

---

**CAUTION** Unless you want to use this feature, if the quiescing period is more than the load balancer polling interval, you must ensure that you do not make any other changes to the `loadbalancer.xml` file before the Application Server instance or the web application has been quiesced. Otherwise, the load balancer will quiesce the Application Server instance or the web application more than once as explained earlier in the section.

---

## Using Multiple Clusters for Online Upgrades Without Loss of Service

You can use the load balancer and multiple clusters to upgrade components within the Sun Java System Application Server without any loss of service. A component can, for example, be a JVM, the Sun Java System Application Server, or a web application.

To achieve this, do the following:

1. Stop one of the clusters. For more information on stopping a cluster, see [“Stopping a Cluster” on page 454](#).
2. Upgrade the component in that cluster.
3. Start the cluster. For more information on starting a cluster, see [“Starting a Cluster” on page 450](#).
4. Repeat the process with the other clusters, one by one.

Because sessions within one cluster will never failover to sessions within another cluster, there is no risk of version mismatch caused by a session failing over from an Application Server instance that is running one version of the component failing over to another Application Server instance (in a different cluster) that is running a different version of the component. A cluster in this way acts as a safe boundary for session failover for the Application Server instances within it.

---

**NOTE** This approach is not possible in the following cases:

- When you change the schema of the high-availability database (HADB). For more information, see [Chapter 21, “Administering the High-Availability Database \(Enterprise Edition\).”](#)
- When you perform an application upgrade that involves a change to the application database schema.

---

**CAUTION** You must upgrade all instances in a cluster together. Otherwise, there is a risk of version mismatch caused by a session failing over from one Application Server instance to another where the Application Server instances have different versions of components running.

---

## Reconfiguring an Application Server Instance in a Running Cluster

There may be cases where you want to make changes to the configuration of an (enabled) Application Server instance that require restarting the Application Server instance. For example, you may want to add a JDBC resource to the Application Server instance, which will necessitate the restarting of the Application Server instance. In such cases, follow these steps:

1. Split the cluster into two clusters with equally distributed Application Server instances by editing the `loadbalancer.xml` file. There should be a minimum of two listeners configured for each cluster (for failover).
2. Disable an Application Server instance in one of the clusters by setting the value of the `enabled` attribute of the `instance` element to `false` in the `loadbalancer.xml` file.
3. After the quiescing period is over, plus a sufficient time to allow the requests being served by the Application Server instance to complete, reconfigure or upgrade the Application Server instance.
4. Restart the Application Server instance.
5. Enable the Application Server instance by setting the value (in the `loadbalancer.xml` file) of the `enabled` attribute of the `instance` element to `true`.
6. Follow steps 2 to 5 for all the Application Server instances in both clusters until all instances are reconfigured.
7. Join the two clusters back to form a single cluster to return to your original configuration.

## Using Multiple Load Balancers

To improve fault tolerance and scalability you can configure a set of clusters using multiple load balancers. You can also have multiple load balancers (on multiple web servers) serving a single Application Server cluster.

All the load balancers must have identical configurations and the `loadbalancer.xml` files for all the load balancers must be identical. It is recommended that you have a master copy of the `loadbalancer.xml` file and use a script to distribute this master copy to all the load balancers in your system. This ensures that the configuration change is done simultaneously to all the configured load balancers. This is particularly useful when enabling or disabling clusters, instances, and applications.

Typically, requests are sent to these load balancers through a single source, for example a hardware load balancer or a DNS-based load distribution mechanism.

Multiple load balancers can also be used for serving two entirely different set of clusters. For example, you might want your secure (HTTPS) applications to be handled by one load balancer and the rest of the applications by another load balancer. In this case, the `loadbalancer.xml` files for these load balancers will be different.

# Configuring Session Persistence (Enterprise Edition)

This chapter describes how to configure session persistence for stateful session beans (SFSBs) in the Sun Java™ System Application Server Enterprise Edition 7 environment.

This chapter includes the following sections:

- [About Session Persistence](#)
- [Basic Session Persistence Configuration Steps](#)
- [Setting Up Availability](#)
- [Failover of J2EE Object References Stored in Sessions](#)
- [Configuring SFSB Session Persistence](#)
- [Configuring HTTP Session Persistence](#)

## About Session Persistence

As an application session proceeds, there is often data that is part of a session that is not stored in a traditional database. An example of such data is the content of a shopping cart. Sun Java System Application Server provides the capability to save, or persist, this session data in a repository so that if an application server instance experiences a failure, the session state can be recovered and the session can continue without information loss.

In J2EE applications, session data is typically stored in HTTP sessions or SFSB sessions. Sun Java System Application Server supports persistence of the state of both HTTP sessions and SFSB sessions. Failover of certain J2EE object references that are stored in both HTTP sessions and SFSB sessions is also supported; see [Table 20-3 on page 478](#).

The high-availability database (HADB) bundled with the Sun Java System Application Server works as the persistence store to provide high availability for session data.

---

**NOTE**      [Appendix B, “Failover Scenarios \(Enterprise Edition\)”](#) describes typical failover scenarios in detail.

---

## Basic Session Persistence Configuration Steps

For successful session persistence configuration, ensure that these steps are followed in the order in which they are presented because some later steps have one or more previous steps as prerequisites.

---

**NOTE**      Some of these steps can be performed using the `clsetup` and `cladmin` commands.

The `clsetup` command is used to automate the process of setting up a basic cluster in a typical configuration. For more information on using the `clsetup` command, see the *Sun Java System Application Server Installation Guide*. The `cladmin` command is used to streamline the process of configuring and administering application server instances in a cluster. For more information on using the `cladmin` command, see [Appendix F, “Using the cladmin Command for Administration \(Enterprise Edition\)”](#) and [“Using the cladmin Command to Configure Session Persistence for an Entire Cluster” on page 485](#).

---

1. Create an HADB database for the cluster. This is explained in [Chapter 21, “Administering the High-Availability Database \(Enterprise Edition\)” on page 507](#).



2. Configure the JDBC connection pool settings and specify a JDBC datasource for the cluster so that the session states for the applications deployed to the cluster can be stored correctly. This is explained in [“Setting Up the JDBC Connection Pool” on page 522](#).
3. Specify the JNDI name of the JDBC resource that you created for the HADB database. This is explained in [“Referencing the HADB Database’s JDBC Resource” on page 467](#).
4. Create a session store for the session data for the HADB database for each cluster. This is explained in [“Creating the Session Store” on page 468](#).
5. Decide if you are using a single cluster or multiple clusters with a single session store. This is explained in [“Managing the Session Store for Clusters” on page 472](#).
6. Enable availability for the application server instances, web or EJB containers, applications, and modules that should support session persistence. This is explained in [“Enabling and Disabling Availability” on page 474](#).
7. Enable availability for any specific SFSBs that require it, and select methods for which checkpointing the session state is necessary. This is explained in the *Developer’s Guide to Enterprise JavaBeans Technology*.
8. If you are *not* enabling availability, you can change the file system session store for SFSBs if desired. This is explained in [“Setting the SFSB Session Store When Availability is Disabled” on page 480](#).
9. Depending on whether you are using a production system and whether you need failover capabilities, specify the persistence type for application server instances or applications. Persistence type essentially defines how and where HTTP session data is stored. This is explained in [“Setting the Persistence Type” on page 485](#).
10. If you are using the `ha` persistence type for HTTP sessions, specify the persistence frequency (how frequently the session is stored) and the persistence scope (how much of the session is stored) for application server instances or web applications. This is explained in [“Setting the Persistence Frequency” on page 492](#) and [“Setting the Persistence Scope” on page 496](#).
11. If you want to fine-tune the HTTP session persistence configuration, modify additional properties as needed (for example, configure the number of seconds between checks for expired sessions). This is explained in [“Configuring Other HTTP Session Persistence Properties” on page 502](#).
12. Make distributable each application that should be highly available. This is explained in [“Making a Web Application Distributable” on page 481](#).

13. Restart each server instance.

---

**NOTE** If the instance is currently serving requests, before you restart the instance, you should first quiesce the instance so that the instance gets enough time to serve the requests it is handling. For more information, see [“Disabling and Quiescing an Application Server Instance in a Cluster” on page 456](#) and [“Stopping an Application Server Instance in a Cluster” on page 453](#).

---

## Setting Up Availability

To set up availability, you must reference the HADB database, create a session store, and enable availability. This section includes the following topics:

- [Configuring JDBC Parameters for the HADB Database](#)
- [Referencing the HADB Database's JDBC Resource](#)
- [Creating the Session Store](#)
- [Clearing the Session Store](#)
- [Managing the Session Store for Clusters](#)
- [Enabling and Disabling Availability](#)

## Configuring JDBC Parameters for the HADB Database

You must configure the JDBC connection pool settings and specify a JDBC datasource for the HADB database so that the session state for the application server instances in each cluster can be stored correctly.

For information on how to do this, see [“Setting Up the JDBC Connection Pool” on page 522](#).

## Referencing the HADB Database's JDBC Resource

While configuring session persistence, you must specify the JNDI name of the JDBC resource that you create for the HADB database. If you use HADB as the persistence store, this information is used to connect to the HADB database.

---

<b>NOTE</b>	If you have used the <code>configure-session-persistence</code> command to configure the session persistence, you would have already specified the JNDI name for the <code>--store</code> option and you do not need to specify it again.
-------------	---

---

To specify the JNDI name of the JDBC resource that you create for the HADB database using the Administration interface:

1. In the left pane of the Administration interface, under the application server instance, open the Containers component.
2. Click the Availability Service tab.
3. Click Properties under Persistence Store Properties.
4. In the Name field, type the following:  
`store-pool-jndi-name`
5. In the Value field, type the JNDI name of the JDBC Resource that you created for the HADB database while configuring the JDBC parameters for the HADB database. For more information, see [Chapter 21, “Administering the High-Availability Database \(Enterprise Edition\).”](#)
6. Click on the OK button.
7. Go to the server instance page.
8. Apply Changes and restart the server.

To specify the JNDI name of the JDBC resource using the command-line interface, set the value of the `store-pool-jndi-name` property for the instance to the value of the JNDI name of the JDBC URL as shown in the example here:

*Example: Specifying the JNDI Name of the JDBC Resource for HADB*

```
set instance1.availability-service.persistence-store.property.store-pool-jndi-name=jdbc_ha
```

In this example, `instance1` is the name of the application server instance and `jdbc_ha` is the value of the JNDI name of the JDBC URL as that you created while configuring the JDBC parameters for the HADB database. For more information on configuring the JDBC parameters, see [“Configuring the HADB” on page 514](#).

## Creating the Session Store

The `asadmin create-session-store` command creates a session store to be used by a particular database user. The syntax is as follows:

```
asadmin create-session-store [--storeurl persistent-store-url --storeuser username
[--storepassword userpassword] [--dbssystempassword systempassword]] |
[--optionsfile file-name]
```

---

**NOTE** The `asadmin create-session-store` command communicates with the HADB database and not with an application server instance. For the `create-session-store` command to work correctly, the HADB database must be running. You must run the `create-session-store` command before accessing any application that needs failover support in a clustered environment.

You can use the `cladmin` command to create a session store for an entire cluster. See [“Using the cladmin Command to Configure Session Persistence for an Entire Cluster” on page 485](#).

The `clsetup` command also configures session persistence. See the *Sun Java System Application Server Installation Guide*.

---

The options of the `create-session-store` command are as follows:

- `storeurl` is the URL of the HADB store. To get this URL, see [“Getting the JDBC URL” on page 522](#).
- `storeuser` is the name of the HADB user that has access to the session store. You can provide any name except the name of the HADB system user. The `create-session-store` command creates this HADB user and gives this user access to the session store.

This is the user name that the application server instance uses to connect to the HADB database through JDBC. Therefore, this is the user name that you must specify when creating the JDBC connection pool for the HADB.

- `storepassword` is the password for the `storeuser`. The `create-session-store` command assigns this password to the `storeuser` when it creates the `storeuser`.
- `dbssystempassword` is the password for the HADB system user. This is the same password that you specify when you create the HADB database using the command `hadbm create`.

For more information on using HADB commands, see [Chapter 21, “Administering the High-Availability Database \(Enterprise Edition\)”](#).

For more information on command syntax, see the command-line interface help.

For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface”](#).

You can also specify `storeurl`, `storeuser`, `storepassword`, and `dbssystempassword` in a file with their corresponding environment variables. You can use the name of this file as the value for the `optionsfile` option.

The environment variables are listed in the following table. In the table, the left column specifies the option and the right column specifies the corresponding environment variable.

**Table 20-1** Environment Variables for the `create-session-store` Command Options

Option	Environment Variable
<code>storeurl</code>	<code>AS_ADMIN_STOREURL</code>
<code>storeuser</code>	<code>AS_ADMIN_STOREUSER</code>
<code>storepassword</code>	<code>AS_ADMIN_STOREPASSWORD</code>
<code>dbssystempassword</code>	<code>AS_ADMIN_DBSYSTEMPASSWORD</code>

These environment variables can be set using the `asadmin set` command. For more information on using the `asadmin set` command, see [Appendix A, “Using the Command Line Interface”](#).

In general, you should be careful while using environment variables for usernames and passwords, as these can be viewed by other users relatively easily. A safer approach is to provide the values for these environment variables in a file and correctly set the permissions on this file. Using a file to specify these values can be useful because in this case others cannot find out these values using, for example, commands that give details of the currently running processes.

The `optionsfile` can exist anywhere on your machine. The file must be a standard text file with each `name=value` pair on a different line. You can specify the location of this file as shown in the example later in the section. Here is an example of `name=value` pairs specified in a file.

```
AS_ADMIN_STOREURL = jdbc:sun:hadb:host1:4045,host2:4085
AS_ADMIN_STOREUSER = MyUser
AS_ADMIN_STOREPASSWORD = MyPassword
AS_ADMIN_DBSYSTEMPASSWORD = SuperUserPassword
```

### *Example: Creating a Session Store*

The following command creates a session store where `storeurl` is `jdbc:sun:hadb:myhost1:4045,myhost2:4085`, `storeuser` is `haadmin`, `storepassword` is `hapassword`, and `dbssystempassword` is `dbhapassword`.

```
asadmin create-session-store --storeurl
jdbc:sun:hadb:myhost1:4045,myhost2:4085 --storeuser haadmin --storepassword
hapassword --dbssystempassword dbhapassword
```

### *Example: Creating a Session Store by Specifying Options in a File*

The following command creates a session store as per the options defined in the `/space/hadetails.txt` file.

```
asadmin create-session-store --optionsfile /space/hadetails.txt
```

## Clearing the Session Store

Sometimes you may want to clear the sessions for all application server instances from the HADB database. This may be needed for example when the application server instance has shut down abruptly and there are many sessions in the HADB database that contain data no longer required. To clear all the sessions from the HADB database, use the `asadmin clear-session-store` command.

---

**CAUTION** The `clear-session-store` command removes all sessions, including passivated sessions, from the HADB database.

---

This section contains the following topics:

- [Before You Clear the Session Store](#)
- [Using the `asadmin clear-session-store` Command](#)

- [After You Clear the Session Store](#)

## Before You Clear the Session Store

You should not use the `asadmin clear-session-store` command if any application server instances in the cluster are persisting session data to the HADB database.

Therefore, before using the `asadmin clear-session-store` command, do one of the following:

- Disable high availability for all instances in the cluster as explained in [“Enabling and Disabling Availability” on page 474](#).
- Quiesce all instances in the cluster as explained in [“Disabling and Quiescing an Application Server Instance in a Cluster” on page 456](#).

## Using the `asadmin clear-session-store` Command

The syntax of the `asadmin clear-session-store` command is as follows:

```
asadmin clear-session-store [--storeurl persistent-store-url] [--storeuser username]
[--storepassword userpassword] [--optionsfile file_name]
```

The values of `storeurl`, `storeuser`, and `storepassword` are the same as explained in [“Creating the Session Store” on page 468](#).

For more information on using HADB commands, see [Chapter 21, “Administering the High-Availability Database \(Enterprise Edition\)”](#).

For more information on command syntax, see the command-line interface help.

For more information on using `asadmin`, see [Appendix A, “Using the Command Line Interface.”](#)

You can also specify `storeurl`, `storeuser` and `storepassword` in a file with their corresponding environment variables. The name of the file can be used as the value for the `optionsfile` option. The environment variables are described in the following table. In the table, the left column specifies the option and the right column specifies the corresponding environment variable.

**Table 20-2** Environment Variables for the `create-session-store` Command Options

Option	Environment variable
<code>--storeurl</code>	<code>AS_ADMIN_STOREURL</code>
<code>--storeuser</code>	<code>AS_ADMIN_STOREUSER</code>
<code>--storepassword</code>	<code>AS_ADMIN_STOREPASSWORD</code>

The file must be a standard text file with each `name=value` pair on a different line. You can specify the location of this file as shown in the example later in the section. Here is an example of `name=value` pairs specified in a file.

```
AS_ADMIN_STOREURL = jdbc:sun:hadb:host1:4045,host2:4085
AS_ADMIN_STOREUSER = MyUser
AS_ADMIN_STOREPASSWORD = MyPassword
```

### *Example: Removing All Sessions from the HADB Database*

In the following example, the `storeurl` is

`jdbc:sun:hadb:myhost1:7676,myhost2:6767`, the `storeuser` is `haadmin`, the `storepassword` is `hapassword`, and the `dbstorepassword` is `dbhapassword`.

```
asadmin clear-session-store --storeurl
jdbc:sun:hadb:myhost1:4045,myhost2:4085 --storeuser haadmin --storepassword
hapassword --dbstorepassword dbhapassword
```

### *Example: Removing All Sessions from the HADB Database by Specifying the Required options in a File*

The following command removes all the sessions in the persistence store according to the options defined in the file `/space/hadetails.txt`.

```
asadmin clear-session-store --optionsfile /space/hadbdetails.txt
```

### After You Clear the Session Store

Once the session is restored, enable availability for all instances in the cluster as explained in [“Enabling and Disabling Availability” on page 474](#).

## Managing the Session Store for Clusters

You can choose whether to use one cluster or multiple clusters with a single session store. For more information on using multiple clusters, see [“Defining Multiple Clusters” on page 449](#).

It is recommended that you *do not* use the same session store for two or more clusters. If, however, you choose to do this, make sure that each cluster is configured with a different cluster name.

You must specify the name of the cluster to which each application server instance belongs.



---

<b>NOTE</b>	If you are using a separate HADB database for each cluster, you do not need to perform this step.
-------------	---

---

For example, if you are using the same session store for two clusters, you can specify the name `cluster1` for instances in the first cluster and the name `cluster2` for instances in the second cluster.

---

<b>NOTE</b>	The name of the cluster need not be the same as the name of the cluster as specified in the <code>loadbalancer.xml</code> file.
-------------	---

---

To specify the name of the cluster for the application server instance using the Administration interface:

1. In the left pane of the Administration interface, under the application server instance, open the Containers component.
2. Click Web Container.
3. Click the Availability Service tab.
4. Click Properties under Persistence Store Properties.
5. In the Name field, type the following:  
`cluster-id`
6. In the Value field, type the name of the cluster for the application server instance.
7. Click on the Save button.
8. Go to the server instance page.
9. Apply Changes and restart the server.

To specify the name of the cluster using the command-line interface, set the value of the `cluster-id` property for the instance to the name of the cluster.

For more information on clusters, see [Chapter 19, “Managing Clusters \(Enterprise Edition\).”](#)

### *Example: Specifying the Cluster Name for an Instance*

In the following example, the cluster name `cluster2` has been specified for an application server instance called `instance1`.

```
set instance1.availability-service.persistence-store.property.cluster-id=cluster2
```

## Enabling and Disabling Availability

Availability can be enabled at five different levels:

1. The server instance
2. The web or EJB container
3. The application
4. The web or EJB module
5. The SFSB

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

The default for a given level is the setting at the next level up. For example, if availability is enabled at the container level, it is enabled by default at the application level.

When availability is disabled at the server instance level (the default setting), enabling it at any other level has no effect. When availability is enabled at the server instance level, it is enabled at all levels unless explicitly disabled.

The following sections describe how to enable availability:

- [Server Instance and Container Levels](#)
- [Application and Module Levels](#)
- [The SFSB Level](#)

## Server Instance and Container Levels

You can enable availability at the server instance or container level in the following ways:

- [Using the Administration Interface](#)
- [Using the asadmin Command](#)
- [Editing the server.xml File](#)

### *Using the Administration Interface*

To enable or disable availability at the server instance or container level using the Administration interface, follow these steps:

1. Open the Availability Service component under your server instance.
2. Go to the Availability Service page.
3. Enable the Instance Level Availability by checking the box. To disable it, uncheck the box.
4. Enable the Web Container Availability by choosing either Enabled or Specified by Instance (if Instance Level Availability is checked). To disable it, choose Disabled.
5. Enable the Ejb Container Availability by choosing either Enabled or Specified by Instance (if Instance Level Availability is checked). To disable it, choose Disabled.
6. Click on the Save button.
7. Go to the server instance page.
8. Apply Changes and restart the server.

### *Using the asadmin Command*

To enable availability at the container level only, use the `asadmin set` command as follows, then restart the server:

```
asadmin set --user admin_user [--password admin_password] [--passwordfile
password_file] [--host localhost] [--port 4848] [--secure | -s]
instance_name.ejb-container.availabilityEnabled=true
```

To enable availability for the web container, use `web-container` instead of `ejb-container`.

*Editing the server.xml File*

To enable availability for the server instance, web container, or EJB container, add `availability-enabled="true"` to the appropriate element of the `server.xml` file as follows, then restart the server:

```
<server name="server1">
    ...
    <availability-service availability-enabled="true">
        ...
    </availability-service>
    ...
    <web-container ... availability-enabled="true"/>
    ...
    <ejb-container ... availability-enabled="true"/>
    ...
</server>
```

**Application and Module Levels**

You can enable availability at the application or module level during deployment in the following ways:

- [Using the Administration Interface](#)
- [Using the `asadmin` Command](#)
- [Editing the `server.xml` File](#)

These sections briefly describe how the deployment procedure is different when you enable availability. For general information about deployment, see [Chapter 13](#), “Deploying Applications.”

### *Using the Administration Interface*

To enable availability in the Administration interface during deployment, make sure Availability Enabled is set to either true or Specified by Container (if availability is enabled for the web or EJB container).

### *Using the asadmin Command*

To enable availability using the `asadmin deploy` or `asadmin deploydir` command, include the `--availabilityenabled` option.

### *Editing the server.xml File*

To enable availability at the application level, set `availability-enabled="true"` in the `j2ee-application` element of the `server.xml` file as follows, then restart the server:

```
<server name="server1">
  ...
  <applications>
    ...
    <j2ee-application
      name="MyApp"
      location="instance_dir/applications/j2ee-apps/MyApp"
      availability-enabled="true">
    </j2ee-application>
    ...
  </applications>
  ...
</server>
```

To enable availability at the web or EJB module level, set `availability-enabled="true"` in the `web-module` or `ejb-module` element of the `server.xml` file as follows, then restart the server:

```
<server name="server1">
  ...
  <applications>
    ...
    <ejb-module
      name="MyBean"
      location="instance_dir/applications/j2ee-modules/MyBean"
      availability-enabled="true">
    </ejb-module>
    ...
  </applications>
  ...
</server>
```

### The SFSB Level

For information about enabling availability for a specific SFSB, see the *Developer's Guide to Enterprise JavaBeans Technology*.

## Failover of J2EE Object References Stored in Sessions

Sun Java System Application Server supports failover of J2EE object references that are stored in HTTP and SFSB Sessions. [Table 20-3 on page 478](#) provides a list of the supported J2EE object references.

The following conditions apply to the recovery of the J2EE object references:

- The recovery can be done only for requests received from web clients. There is no support for failover for requests received from application clients.
- All user-defined objects bound into the session must implement the `java.io.Serializable` interface or belong to one of the supported J2EE object references types.

If an object neither implements `java.io.Serializable` interface nor belongs to one of the supported J2EE object references types, the object is not persisted. *An exception is generated* and the stack trace is written to the server log.

- A reference to a J2EE component or resource may be lost. [Table 20-3 on page 478](#) lists the types of references that session failover supports.
- References to open files or network connections are lost.

For information about how to work around some of these limitations, see *Sun Java System Application Server System Deployment Guide*.

**Table 20-3** Object Types Supported for Session State Failover

Java Object Type	Failover Support
EntityBean local home reference, local object reference	Yes
Stateful SessionBean local home reference	Yes
Stateful SessionBean local object reference	Yes
Stateless SessionBean local home reference, local object reference	Yes

**Table 20-3** Object Types Supported for Session State Failover (*Continued*)

Java Object Type	Failover Support
Co-located EntityBean remote home reference, remote reference	Yes
Co-located Stateful SessionBean remote home reference	Yes
Co-located Stateful SessionBean remote reference	Yes
Co-located Stateless SessionBean remote home reference, remote reference	Yes
Distributed EntityBean remote home reference, remote reference	Yes
Distributed Stateful SessionBean remote home reference, remote reference	Yes
Distributed Stateless SessionBean remote home reference, remote reference	Yes
JNDI Context	Yes, InitialContext and <code>java:comp/env</code>
UserTransaction	Yes, but if the instance that fails is never restarted, any prepared global transactions are lost and may not be correctly rolled back or committed
JDBC DataSource	No
Java™ Message Service (JMS) ConnectionFactory, Destination	No
JavaMail™ Session	No
Connection Factory	No
Administered Object	No
Web service reference	No
Serializable Java types	Yes

## Configuring SFSB Session Persistence

This section includes the following topics:

- [How SFSB Session Persistence Works When Availability Is Enabled](#)
- [Setting the SFSB Session Store When Availability is Disabled](#)

## How SFSB Session Persistence Works When Availability Is Enabled

When availability is enabled at the EJB container and application levels, SFSB session persistence is enabled by default. See [“Enabling and Disabling Availability” on page 474](#).

If availability is enabled, the state of an SFSB is saved to the persistent store at predefined points in its life cycle. This is called *checkpointing*. Checkpointing generally occurs after any transaction involving the SFSB is completed, even if the transaction rolls back. You can also configure end-of-method checkpointing.

For more information about SFSB checkpointing, see the *Developer’s Guide to Enterprise JavaBeans Technology*.

## Setting the SFSB Session Store When Availability is Disabled

If availability is disabled, the local file system is used for SFSB state passivation, but not persistence. To change where the SFSB state is stored, follow these steps in the Administration interface:

1. Go to the server instance page.
2. Click on the Advanced tab.
3. Edit the Session Store Location value.
4. Click on the Save button.
5. Go to the server instance page.
6. Apply Changes and restart the server.

As an alternative, you can edit the `server.xml` file. The `session-store` attribute 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">
```

This change to the `server.xml` file take effect when you restart the server.

Finally, you can use the `asadmin set` command. For example:

```
asadmin set --user joeuser --password secret server1.sessionStore=/export/sfsbstore
```

This change takes effect when you restart the server.



# Configuring HTTP Session Persistence

This section includes the following topics:

- [Making a Web Application Distributable](#)
- [Availability of Single Sign-on in the HTTP Session State](#)
- [Setting the Session Timeout](#)
- [HTTP Session Persistence Options](#)

## Making a Web Application Distributable

For HTTP session persistence to work for a web application, the web application must be *distributable*. Configuring a non-distributable web application for high availability has no effect. For details, see the *Sun Java System Application Server Developer's Guide to Web Applications*.

To be distributable, an application must also conform to the behavior and rules prescribed for a distributable application as per Java Servlet Specification 7.7.2. For more information, see *Java Servlet Specification, v2.3*.

## Availability of Single Sign-on in the HTTP Session State

In a single application server instance, once a user is authenticated by an application, the user is not required to reauthenticate individually to other applications running on the same instance. This is called *single sign-on*. For more information on single sign-on, see [“Web Application Deployment” on page 183](#).

For this feature to continue to work even when an HTTP session fails over to another instance in a cluster, single sign-on information must be persisted to the HADB. This is enabled when you enable availability for the application server instance and the web container.

### Session Availability for Applications Belonging to a Single Sign-On Group

Applications that can be accessed through a single name and password combination constitute a *single sign-on group*.

For HTTP sessions corresponding to applications that are part of a single sign-on group, if one of the sessions times out, other sessions are not invalidated and continue to be available. This is because time out of one session should not affect the availability of other sessions.

As a corollary of this behavior, if a session times out and you try to access the corresponding application from the same browser window that was running the session, you are not required to authenticate again. However, a new session is created.

Take the example of a shopping cart application that is a part of a single sign-on group with two other applications. Assume that the session time out value for the other two applications is higher than the session time out value for the shopping cart application. If your session for the shopping cart application times out and you try to run the shopping cart application from the same browser window that was running the session, you are not required to authenticate again. However, the previous shopping cart is lost, and you will have to create a new shopping cart. The other two applications continue to run as usual even though the session running the shopping cart application has timed out.

Similarly, suppose a session corresponding to any of the other two applications times out. You are not required to authenticate again while connecting to the application from the same browser window in which you were running the session.

---

<b>NOTE</b>	This behavior applies only to cases where the session times out. If single sign-on is enabled and you invalidate one of the sessions using <code>HttpSession.invalidate()</code> , the sessions for all applications belonging to the single sign-on group are invalidated. If you try to access any application belonging to the single sign-on group, you are required to authenticate again, and a new session is created for the client accessing the application.
-------------	--

---

## Setting the Session Timeout

HTTP sessions are persisted for the interval set in the `session-timeout` element in the `web.xml` file or the `timeoutSeconds` property of the `session-properties` element in the `server.xml` file (for instance-level configuration) or in the `sun-web.xml` file (for application-level configuration). Note that the `session-timeout` element in `web.xml` is specified in minutes, not seconds.

If a `session-timeout` element is specified, the `session-timeout` value overrides any `timeoutSeconds` value. If `timeoutSeconds` is specified in both `sun-web.xml` and `server.xml`, the value in `sun-web.xml` takes precedence. If neither `session-timeout` nor `timeoutSeconds` is specified, the `timeoutSeconds` default of 600 seconds (10 minutes) is used.

If `session-timeout` is not specified and `timeoutSeconds` is set to 0, HTTP sessions never time out.

Inactive sessions that have timed out are deleted from the session store at the frequency specified by 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). If `reapIntervalSeconds` is specified in both `sun-web.xml` and `server.xml`, the value in `sun-web.xml` takes precedence. The default is 60 seconds.

To maximize availability, set `session-timeout` or `timeoutSeconds` to a large value, and set `reapIntervalSeconds` to 1, the minimum value.

## HTTP Session Persistence Options

You can configure HTTP session persistence for the Sun Java System Application Server to balance your particular needs for performance, reliability, and high availability. For example, every time your data is saved, you can store the entire session, or you can store the session only if the session has been modified. Or, you can configure the persistence setting so that only the modified attributes of the session are stored. Similarly you can choose whether the session is stored at the end of every web request (thus providing high availability and reliability of updated session state), or you can choose to store the session after a time interval you specify (thus providing better performance).

You can specify the instance-level settings through the Administration interface or through the command-line `configure-session-persistence` command.

If you want an application to have its own session persistence settings, you can override the container-level settings by setting the corresponding configuration for the application in the corresponding `sun-web.xml` file.

---

**NOTE** The session persistence configuration must be identical for a given application across all instances of a cluster. Do not deploy an application with one set of settings to one segment of instances in a cluster and another set of settings to another segment of instances in the cluster.

---

This section includes the following topics:

- [About the asadmin configure-session-persistence Command](#)
- [Using the cladmin Command to Configure Session Persistence for an Entire Cluster](#)
- [Setting the Persistence Type](#)
- [Comparison of Persistence Type Options](#)
- [Setting the Persistence Frequency](#)
- [Comparison of Persistence Frequency Options](#)
- [Setting the Persistence Scope](#)
- [Comparison of Persistence Scope Options](#)
- [Configuring Other HTTP Session Persistence Properties](#)
- [Overriding HTTP Session Persistence Settings](#)

### About the asadmin configure-session-persistence Command

To specify the session persistence options for the application server instance, in addition to using the Administration interface, you can use the `configure-session-persistence` command. The syntax of the `asadmin configure-session-persistence` command is as follows:

```
asadmin configure-session-persistence standard-options [--type
persistence-type] [--frequency frequency] [--scope scope] [--store
jdbc-resource-jndi-name] [--property name=value[:name=value]*] instance_name
```

- `type` is the persistence type that defines the location where session data is stored. The *persistence-type* is explained in the section [“Setting the Persistence Type” on page 485](#).
- `frequency` is the persistence frequency at which the session is stored. Persistence frequency is explained in the section [“Setting the Persistence Frequency” on page 492](#).
- `scope` is the amount of session that is stored. Persistence scope is explained in the section [“Setting the Persistence Scope” on page 496](#).
- `store` is the JNDI name of the JDBC resource that you specified for the HADB database. For more information, see [“Configuring JDBC Parameters for the HADB Database” on page 466](#).

- `property` option is used to fine-tune other properties for configuring session persistence. For more information, see [“Configuring Other HTTP Session Persistence Properties” on page 502](#).
- `instance_name` is the name of the application server instance for which you are configuring the session persistence.

For more information on command syntax, see the command-line interface help.

For more information on using the `asadmin` command, see [Appendix A, “Using the Command Line Interface.”](#)

## Using the `cladmin` Command to Configure Session Persistence for an Entire Cluster

As an alternative to the `asadmin configure-session-persistence` command, you can use the `cladmin` command to configure session persistence for all instances in a cluster at once as follows:

```
cladmin [--instancefile instance_file_location] [--passwordfile password_file_location]
configure-session-persistence [--type persistence-type] [--frequency frequency]
[--scope scope] [--store jdbc-resource-jndi-name] [--property name=value[:name=value]*]
```

where:

- `instance_file_location` is the location of the `clinstance.conf` file
- `password_file_location` is the location of the `clpassword.conf` file

If the `clinstance.conf` and the `clpassword.conf` files are located in the configuration directory of the Sun Java System Application Server (by default `etc/opt/SUNWappserver7`), you can omit the `instancefile` and `passwordfile` options.

For more information on the `cladmin` command, see [Appendix F, “Using the `cladmin` Command for Administration \(Enterprise Edition\).”](#)

## Setting the Persistence Type

The *persistence type* defines if and how HTTP session persistence and failover take place in the Sun Java System Application Server. Persistence types are `ha`, `memory`, and `file`.

When you set the persistence type to `ha`, the HADB is used as the persistence store. The persistence type `ha` is the persistence type supported for production environments that require session persistence and failover capabilities.

The `memory` persistence type provides no failover and is the default persistence type for HTTP sessions. It is the best choice for production environments that *do not* require HTTP session persistence and failover capabilities.

The `persistence type` setting defines the location where the session state is stored. You can choose one of the three options available:

- `ha`
- `memory`
- `file`

Information about these persistence types is provided in the subsequent sections. A comparison of the persistence types and the configuration options that they support is provided in [“Comparison of Persistence Type Options” on page 492](#).

### *Default Values for Session Persistence*

If high availability is disabled, persistence type is set to `memory` for HTTP sessions and SFSB sessions use the SFSB session store. If high availability is enabled, persistence type is set to `ha`. If no further configuration exists either at the instance-level or at the application level, then the default session persistence configuration is as follows:

- Persistence type is set to `ha`.
- Persistence frequency is set to `web-method`.
- Persistence scope is set to `session`.

This default can be overridden by any other persistence configuration, either for an application server instance or for an application.

### *Setting the Persistence Type to ha*

The persistence type `ha` lets you store session data in the HADB database. This is the persistence type that you must use to enable failover of session state between application server instances in a cluster. This is the only persistence type supported for production systems *that require HTTP session persistence and failover of session state*.

In this persistence type, the data for sessions being serviced by server instances in a cluster is stored in the HADB database. The state of each session is available to all instances in the cluster. Therefore, if an instance becomes unavailable, other instances in the cluster can continue to serve the sessions that the unavailable instance was serving.

Here are the details of what happens when an application server instance in a cluster becomes unavailable.

- The load balancer keeps monitoring all application server instances to check if they are available. If an instance becomes unavailable, the load balancer marks the instance as unavailable. For more information on monitoring, see [“Monitoring the Load Balancer Plug-in” on page 429](#).
- The load balancer stops sending any unassigned requests to the unavailable instance. An unassigned request is a request for a new session. For more information, see [“Defining a Cluster” on page 445](#).
- Idempotent URLs are supported along the HTTP path, but not the RMI/IIOP path. If a server instance to which an RMI/IIOP client request is sent crashes during the request processing (before the response could be prepared and sent back to the client), an error is propagated to the client. The client must retry the request explicitly. When the client retries the request, the request is sent to another server instance in the cluster, which retrieves session state information for this client.
- All subsequent requests for sessions assigned to the application server instance are failed over to other instances in the cluster.

For more information about clusters, see [Chapter 19, “Managing Clusters \(Enterprise Edition\)”](#).

When you use the `ha` persistence type, you must also correctly configure the persistence frequency (to specify the frequency at which the session data is stored) and the persistence scope (to specify how much of the session is stored). For more information on setting the persistence frequency, see [“Setting the Persistence Frequency” on page 492](#). For more information on setting the persistence scope, see [“Setting the Persistence Scope” on page 496](#).

To set the persistence type to `ha` for an application server instance using the Administration interface:

1. In the left pane of the Administration interface, under the application server instance, open the Containers component.
2. Click Web Container.
3. Click the Session Manager tab.
4. From the drop-down list under General, select Highly Available Database.
5. Click on the Save button.
6. Go to the server instance page.

## 7. Apply Changes and restart the server.

To set the persistence type to `ha` using the command-line interface, specify the value `ha` for the `type` option of the `configure-session-persistence` command.

### Example: Setting the Persistence Type to `ha` for an Instance

```
asadmin configure-session-persistence --user admin --password MyPassword
--host localhost --port 4848 --type ha --frequency web-method --scope
modified-session --store jdbc_hastore instance1
```

To set the persistence type to `ha` for an application, in the `sun-web.xml` file, set the value of the attribute `persistence-type` of the element `session-manager` to `ha`.

### Example: Setting the Persistence Type to `ha` for an Application

The example here provides extracts from a sample `sun-web.xml` file. The relevant portion has been **highlighted**.

```
<session-manager persistence-type="ha">
  <manager-properties>
    <property name="persistenceFrequency" value="web-method"/>
  </manager-properties>
  <store-properties>
    <property name="persistenceScope" value="session"/>
  </store-properties>
</session-manager>
```

### *Setting the Persistence Type to memory*

If availability for the application server instance is disabled, then `memory` is the default persistence type.

The `memory` persistence type provides no session persistence in a clustered environment.

However, for a single application server instance, you can configure the `memory` persistence type so that before a regular shutdown of the instance, all session state is written to a file. To enable this, you must specify the directory in which you want the session state to be stored in case of shutdown. You specify the directory by providing the path of the directory as the value for the `sessionFilename` property of `session-manager` in the `server.xml` file. For more information on setting the value of `sessionFilename`, see [“Configuring Other HTTP Session Persistence Properties” on page 502](#).



If the application server instance becomes unavailable unexpectedly, the corresponding session state that would be stored in the file is lost, and therefore, in case of unexpected shutdown of an application server instance, the instance cannot recover the session state.

---

**NOTE** The limited failover capabilities offered by the `memory` persistence type are intended for use only in development systems. The `memory` persistence type is not intended—and not supported—for use in production systems where HTTP session persistence and failover capabilities are required.

However, the `memory` persistence type is the best choice for production environments that *do not* require HTTP session persistence and failover capabilities.

---



---

**NOTE** You cannot choose the configuration of persistence frequency and persistence scope for the `memory` persistence type.

---

To set the persistence type to `memory` for an application server instance using the Administration interface:

1. In the left pane of the Administration interface, under the application server instance, open the Containers component.
2. Click Web Container.
3. Click the Session Manager tab.
4. From the drop-down list under General, select `memory`.
5. Click on the Save button.
6. Go to the server instance page.
7. Apply Changes and restart the server.

To set the persistence type to `memory` for an application server instance using the command-line interface, specify the value `memory` for the `type` option of the `configure-session-persistence` command.

**Example: Setting the Persistence Type to `memory` for an Instance**

```
asadmin configure-session-persistence --user admin --password
Mypassword --host localhost --port 4848 --type memory instance1
```

To set the persistence type to `memory` for an application, in the `sun-web.xml` file for the application, set the value of the attribute `persistence-type` of the element `session-manager` to `memory`.

**Example: Setting the Persistence Type to memory for an Application**

The example here provides extracts from a sample `sun-web.xml` file. The relevant portion has been **highlighted**.

```
<session-manager persistence-type="memory">
</session-manager>
```

*Setting the Persistence Type to file*

The `file` persistence type provides no session persistence in a clustered environment.

For an application server instance, you can use the `file` persistence type to store session state in a file. The Sun Java System application server stores information for each session in a separate file. You do not have to specify the path of these files. You can change the path where these files are written. For more information, see [Table 20-8 on page 503](#).

If the instance becomes unavailable and restarts, it can recover the session state that was last written to the files.

The session state is stored periodically to the files when the background thread that triggers the storage runs. This means that you may experience some loss of session data when a graceful shutdown or unexpected shutdown of the instance occurs.

You can configure the frequency at which the background thread runs by setting 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). For more information, see [“Configuring Other HTTP Session Persistence Properties” on page 502](#).

---

<b>NOTE</b>	The limited failover capabilities offered by the <code>file</code> persistence type are intended for use only in development systems. The <code>file</code> persistence type is not intended—and not supported—for use in production systems where HTTP session persistence and failover capabilities are required.
-------------	---

---

A possible use of the `file` persistence type is in a development environment to simulate failover during development without having to run HADB.

---

**NOTE** You cannot choose the configuration of persistence frequency and persistence scope for the `file` persistence type.

---

To set the persistence type to `file` for an application server instance using the Administration interface:

1. In the left pane of the Administration interface, under the application server instance, open the Containers component.
2. Click Web Container.
3. Click the Session Manager tab.
4. From the drop-down list under General, select `file`.
5. Click on the Save button.
6. Go to the server instance page.
7. Apply Changes and restart the server.

To set the persistence type to `file` using the command-line interface, specify the value `file` for the `type` option of the `configure-session-persistence` command.

**Example: Setting the Persistence Type to file for an Instance**

```
asadmin configure-session-persistence --user admin --password MyPassword
--host localhost --port 4848 --type file instance1
```

To set the persistence type to `file` for an application, in the `sun-web.xml` file, set the value of the attribute `persistence-type` of the element `session-manager` to `file`.

**Example: Setting the Persistence Type to file for an Application**

The example here provides extracts from a sample `sun-web.xml` file. The relevant portion has been **highlighted**.

```
<session-manager persistence-type="file">
</session-manager>
```

### Comparison of Persistence Type Options

The table here provides a comparison of and the supported combinations for the persistence type option. The leftmost column lists the persistence type option, the next column lists whether this option is intended or supported for failover in production systems, the next column lists the typical use of the option, the next column lists the persistence frequency option(s) that the option supports, and the rightmost column lists the persistence scope option(s) that the option supports.

**Table 20-4** Comparison of Persistence Type Options for HTTP Sessions

Persistence Type Option	Intended or Supported for Failover in Production Systems	Typical Use	Persistence Frequency Supported	Persistence Scope Supported
memory	No	<ul style="list-style-type: none"><li>In development systems</li><li>In production systems where no failover capabilities are needed</li></ul>	Not applicable	Not applicable
file	No	In development systems, including to simulate failover during development without having to run HADB.	time-based	Not applicable
ha	Yes	In production systems where failover capabilities are required.	<ul style="list-style-type: none"><li>time-based</li><li>web-method</li></ul>	<ul style="list-style-type: none"><li>session</li><li>modified-session</li><li>modified-attribute</li></ul>

If there is an invalid configuration, the persistence type is set to `memory` and the application continues to run. A message about the invalid configuration is logged.

For information about SFSB persistence options, see [“Setting the SFSB Session Store When Availability is Disabled” on page 480](#).

### Setting the Persistence Frequency

If you are using HADB to store session state, then depending on your requirements of high availability and performance, you can configure the frequency at which the session state is stored in the HADB database. You must choose one of two options available:

- `web-method`
- `time-based`

---

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

---

### *Setting the Persistence Frequency to web-method*

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

This mode is useful when there is a need for high availability of updated session state. This method provides the highest guarantee that session state will be available after instance failure.

To set the persistence frequency to `web-method` for an application server instance using the Administration interface:

1. In the left pane of the Administration interface, under the application server instance, open the Containers component.
2. Click Web Container.
3. Click the Session Manager tab.
4. Click the Properties button under Properties.
5. In the Name field, type the following:  
`persistenceFrequency`
6. In the Value field, type the following:  
`web-method`
7. Click on the Save button.
8. Go to the server instance page.
9. Apply Changes and restart the server.

To set the persistence frequency to `web-method` using the command-line interface, specify the value `web-method` for the `frequency` option of the `configure-session-persistence` command.

### **Example: Setting the Persistence Frequency to web-method for an Instance**

```
asadmin configure-session-persistence --user admin --password MyPassword
--host localhost --port 4848 --type ha --frequency web-method --scope
modified-session --store jdbc_hastore instance1
```

To set the persistence frequency to `web-method` for an application, in the `sun-web.xml` file, in `manager-properties`, set the value of the property `persistenceFrequency` to `web-method`.

**Example: Setting the Persistence Frequency to `web-method` for an Application**

The example here provides extracts from a sample `sun-web.xml` file. The relevant portion has been **highlighted**.

```
<session-manager persistence-type="ha">
    <manager-properties>
        <property name="persistenceFrequency"
value="web-method"/>
    </manager-properties>
    <store-properties>
        <property name="persistenceScope" value="session"/>
    </store-properties>
</session-manager>
```

*Setting the Persistence Frequency to 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).

For more information on setting the value of `reapIntervalSeconds`, see [“Configuring Other HTTP Session Persistence Properties” on page 502](#).

The time-based persistence frequency provides a relatively high guarantee (but not as high as the `web-method` frequency type) that session state will be highly available. Because the session is stored after a configurable time interval and not after every web request, a better response time is achieved.

The trade off in using the time-based persistence frequency is that there is a small risk that any changes to session state that occurred since the last time the session was stored will be lost if an instance failure occurs.

To set the persistence frequency to `time-based` for an application server instance using the Administration interface:

1. In the Administration interface, in the left pane, for the application server instance, open the Containers component.
2. Click Web Container.

3. Click the Session Manager tab.
4. Click the Properties button under Properties.
5. In the Name field, type the following:  
persistenceFrequency
6. In the Value field, type the following:  
time-based
7. Click on the Save button.
8. Go to the server instance page.
9. Apply Changes and restart the server.

To set the persistence frequency to `time-based` using the command-line interface, specify the value `time-based` for the `frequency` option of the `configure-session-persistence` command.

**Example: Setting the Persistence Frequency to time-based for an Instance**

```
asadmin configure-session-persistence --user admin --password MyPassword
--host localhost --port 4848 --type ha --frequency time-based --scope
modified-session --store jdbc_hastore instance1
```

To set the persistence frequency to `time-based` for an application, in the `sun-web.xml` file, set the value of the property `persistenceFrequency` of the `manager-properties` element to `time-based`.

**Example: Setting the Persistence Frequency to time-based for an Application**

The example here provides extracts from a sample `sun-web.xml` file. The relevant portion has been **highlighted**.

```
<session-manager persistence-type="ha">
    <manager-properties>
        <property name="persistenceFrequency"
value="time-based"/>
    </manager-properties>
    <store-properties>
        <property name="persistenceScope" value="session"/>
    </store-properties>
</session-manager>
```

### Comparison of Persistence Frequency Options

The table here summarizes the advantages and disadvantages of the persistence frequency options. The left column lists the persistence frequency option, the middle column lists the advantage of the option, and the right column lists the possible disadvantage of the option.

**Table 20-5** Comparison of Persistence Frequency Options

Persistence Frequency Option	Advantage(s)	Disadvantage(s)
web-method	Guarantees that the most up-to-date session state is available.	Potentially increased response time and reduced throughput.
time-based	Better response time and potentially better throughput.	Less guarantee than the web-method persistence frequency that the most updated session state is available after the failure of an application server instance.

### 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. You must choose one of the three options 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 database.
- `modified-attribute`. Only the modified attributes of the session are saved.

These options are described in the following topics:

- [Setting the Persistence Scope to modified-session](#)
- [Setting the Persistence Scope to session](#)
- [Setting the Persistence Scope to modified-attribute](#)

<b>NOTE</b>	You can choose the persistence scope only for the <code>ha</code> persistence type.
-------------	---



### *Setting the Persistence Scope to modified-session*

In the `modified-session` mode, the session is stored only if it has been modified. Thus, if the persistence frequency is `web-method`, the entire session is stored at the end of every web request, just before sending a response back to the client *only if* the session has changed from the last time it was stored. Similarly, if the persistence frequency is `time-based`, the entire session is stored after each time-based frequency that you specify *only if* the session has changed from the last time it was stored.

A session is considered to be modified only if the `setAttribute` method (if the attribute was changed) or the `removeAttribute` method (if the attribute was removed) was invoked on the session during the execution of a web method (typically `doGet` or `doPost`).

To set the persistence scope to `modified-session` for an application server instance using the Administration interface:

1. In the left pane of the Administration interface, under the application server instance, open the Containers component.
2. Click Web Container.
3. Click the Session Manager tab.
4. Click the Properties button under Session Store Properties.
5. In the Name field, type the following:  
`persistenceScope`
6. In the Value field, type the following:  
`modified-session`
7. Click on the Save button.
8. Go to the server instance page.
9. Apply Changes and restart the server.

To set the persistence scope to `modified-session` using the command-line interface, specify the value `modified-session` for the `scope` option of the `configure-session-persistence` command.

#### **Example: Setting the Persistence Scope to modified-session for an Instance**

```
asadmin configure-session-persistence --user admin --password MyPassword
--host localhost --port 4848 --type ha --frequency web-method --scope
modified-session --store jdbc_hastore instance1
```

To set the scope mode to `modified-session` for an application, in the `sun-web.xml` file, set the value of the property `persistenceScope` of the `store-properties` element to `modified-session`.

**Example: Setting the Persistence Scope to `modified-session` for an Application**

The example here provides extracts from a sample `sun-web.xml` file. The relevant portion has been **highlighted**.

```
<session-manager persistence-type="ha">
  <manager-properties>
    <property name="persistenceFrequency" value="web-method"/>
  </manager-properties>
  <store-properties>
    <property name="persistenceScope"
value="modified-session"/>
  </store-properties>
</session-manager>
```

*Setting the Persistence Scope to session*

In the `session` persistence scope, the entire session is stored every time. Thus, if the persistence frequency is `web-method`, the entire session is stored at the end of every web request, just before sending a response back to the client. Similarly, if the persistence frequency is `time-based`, the entire session is stored after each time-based frequency that you specify.

If an application does not call `setAttribute` or `removeAttribute` every time an attribute has changed, the other persistence scope options cannot be used and the persistence scope `session` is the only available option.

To set the persistence scope to `session` for an application server instance using the Administration interface:

1. In the left pane of the Administration interface, under the application server instance, open the Containers component.
2. Click Web Container.
3. Click the Session Manager tab.
4. Click the Properties button under Session Store Properties.
5. In the Name field, type the following:

```
persistenceScope
```

6. In the Value field, type the following:

```
session
```

7. Click on the Save button.
8. Go to the server instance page.
9. Apply Changes and restart the server.

To set the persistence scope to `session` using the command-line interface, specify the value `session` for the `scope` option of the `configure-session-persistence` command.

#### Example: Setting the Persistence Scope to Session

```
configure-session-persistence --user admin --password MyPassword --host
localhost --port 4848 --type ha --frequency web-method --scope session
--store jdbc_hastore instance1
```

To set the scope mode to `session` for an application, in the `sun-web.xml` file, set the value of the `persistenceScope` property of the `store-properties` element to `session`.

#### Example: Setting the Persistence Scope to session for an Application

The example here provides extracts from a sample `sun-web.xml` file. The relevant portion has been highlighted.

```
<session-manager persistence-type="ha">
  <manager-properties>
    <property name="persistenceFrequency" value="web-method"/>
  </manager-properties>
  <store-properties>
    <property name="persistenceScope" value="session"/>
  </store-properties>
</session-manager>
```

#### *Setting the Persistence Scope to modified-attribute*

In the `modified-attribute` persistence scope, only those attributes are stored that have been modified (inserted, updated, or deleted) since the last time the session was stored. Using this mode can improve the throughput and response time significantly for applications for which only a small portion of the session state is modified for any given request.

If you use the `modified-attribute` persistence scope, your application must follow these guidelines:

- Call `setAttribute` or `removeAttribute` every time you modify the session state.
- Make sure there are no cross references between attributes.

This is because the object graphs under each distinct attribute key will be serialized and stored separately and if there are any object cross reference between the objects under each separate key, they will not get serialized and deserialized correctly.

- Distribute the session state across multiple attributes, or at least between a read-only attribute and a modifiable attribute.

To set the persistence scope to `modified-attribute` for an application server instance using the Administration interface:

1. In the left pane of the Administration interface, under the application server instance, open the Containers component.
2. Click Web Container.
3. Click the Session Manager tab.
4. Click the Properties button under Session Store Properties.
5. In the Name field, type the following:  
`persistenceScope`
6. In the Value field, type the following:  
`modified-attribute`
7. Click on the Save button.
8. Go to the server instance page.
9. Apply Changes and restart the server.

To set the persistence scope to `modified-attribute` using the command-line interface, specify the value `modified-attribute` for the `scope` option of the `configure-session-persistence` command.

**Example: Setting the Persistence Scope to `modified-attribute` for an Instance**

```
asadmin configure-session-persistence --user admin --password MyPassword
--host localhost --port 4848 --type ha --frequency web-method --scope
modified-attribute --store jdbc_hastore instance1
```

To set the scope mode to `modified-attribute` for an application, in the `sun-web.xml` file, set the value of the `persistenceScope` property of the `store-properties` element to `modified-attribute`.

#### Example: Setting the Persistence Scope to `modified-attribute` for an Application

The example here provides extracts from a sample `sun-web.xml` file. The relevant portion has been **highlighted**.

```
<session-manager persistence-type="ha">
  <manager-properties>
    <property name="persistenceFrequency" value="web-method"/>
  </manager-properties>
  <store-properties>
    <property name="persistenceScope"
value="modified-attribute"/>
  </store-properties>
</session-manager>
```

## Comparison of Persistence Scope Options

The table here summarizes the advantages and disadvantages of the persistence scope options.

**Table 20-6** Comparison of Persistence Scope Options

Persistence Scope Option	Advantage(s)	Disadvantage(s)
modified-session	Provides improved response time for requests that do not modify session state.	Your application must call the <code>setAttribute</code> method (if the attribute was changed) or the <code>removeAttribute</code> method (if the attribute was removed) on the session during the execution of a web method (typically <code>doGet</code> or <code>doPost</code> ).
session	No constraint on applications.	Potentially degraded throughput and response time as compared to the <code>modified-session</code> and the <code>modified-attribute</code> options.

**Table 20-6** Comparison of Persistence Scope Options *(Continued) (Continued) (Continued)*

Persistence Scope Option	Advantage(s)	Disadvantage(s)
modified-attribute	Better throughput and response time for requests in which the percentage of session state modified is low.	<div><div>1. As the percentage of session state that gets modified for a given request grows to around 60%, the throughput and the response time degrade. In such cases, the performance gets worse than the <code>session</code> or <code>modified-session</code> persistence scope because of the overhead of splitting the attributes into separate records.</div><div>2. Your application must be written to meet the following constraints required by this mode:<ul style="list-style-type: none"><li>• Call <code>setAttribute</code> or <code>removeAttribute</code> every time you modify the session state.</li><li>• Make sure there are no cross references between attributes.</li><li>• Distribute the session state across multiple attributes, or at least between a read-only attribute and a modifiable attribute.</li></ul></div></div>

### Configuring Other HTTP Session Persistence Properties

To fine tune the session persistence behavior, you can modify properties of the `manager-properties`, `store-properties`, and `session-properties` subelements of the `session-manager` element in the `server.xml` file (for instance-level configuration) or in the `sun-web.xml` file (for application-level configuration). You can modify these properties using the `asadmin set` command. These properties are explained here in the tables [Table 20-7](#) and [Table 20-8](#). In the tables, the left column specifies the name of the property, the middle column specifies the default value of the property, and the right column provides a description of the property.

**Table 20-7** `manager-properties` Configurable for Session Persistence

Name of the Property	Default Value	Description
<code>reapIntervalSeconds</code>	60	<p>Specifies the number of seconds between checks for modified or timed-out sessions. This is also the frequency at which passivation of sessions occurs.</p> <p>If the persistence type is <code>file</code> or <code>ha</code>, sessions are passivated if <code>maxSessions</code> has been exceeded.</p> <p>If the persistence frequency is <code>time-based</code>, active sessions are stored at this interval.</p>
<code>maxSessions</code>	-1	<p>Specifies the maximum number of sessions that can be in cache, or -1 for no limit. After this limit is reached:</p> <ul style="list-style-type: none"> <li>• If the persistence type is <code>memory</code>, an attempt to create a new session causes an <code>IllegalStateException</code> to be thrown.</li> <li>• If the persistence-type is <code>file</code> or <code>ha</code>, the sessions are passivated to the persistent store.</li> </ul>
<code>sessionFilename</code>	none; state is not preserved across restarts	<p>Specifies the absolute or relative path to the directory in which the session state is preserved between application restarts, if preserving the state is possible. A relative path is relative to the temporary directory for this application.</p> <p>Applicable only if the persistence-type is <code>memory</code>.</p>

**Table 20-8** `store-properties` Configurable for Session Persistence

Property Name	Default Value	Description
<code>directory</code>	<i>instance_dir/generated/jsp/j2ee-apps</i> <i>/appname/appname_war</i>	<p>Specifies the absolute or relative path to the directory into which individual session files are written. A relative path is relative to the temporary work directory for this application.</p> <p>Applicable only if the persistence type is <code>file</code>.</p>

**Table 20-9** session-properties Configurable for Session Persistence

Property Name	Default Value	Description
timeoutSeconds	600	<p>Specifies the default maximum inactive interval (in seconds) for sessions. If set to 0 or less, sessions never expire.</p> <p>If a session-timeout element is specified in the web.xml file, the session-timeout value overrides any timeoutSeconds value. If timeoutSeconds is specified in both sun-web.xml and server.xml, the value in sun-web.xml takes precedence. If neither session-timeout nor timeoutSeconds is specified, the timeoutSeconds default is used.</p> <p>Note that the session-timeout element in web.xml is specified in minutes, not seconds.</p>

**Example: Setting the maxSessions and reapIntervalSeconds Properties**

In the following example, the value of maxSessions has been set to 1000 and the value of reapIntervalSeconds has been set to 60.

```
asadmin configure-session-persistence --user admin --password MyPassword
--host localhost --port 4848 --type ha --property
maxSessions=1000:reapIntervalSeconds=60 instance1
```

**Overriding HTTP Session Persistence Settings**

For a web application, you can specify the session persistence settings through the corresponding sun-web.xml file.

You may want to specify different session persistence settings for a web application when, for example, the application has specific performance or reliability requirements that other applications deployed on the cluster do not have.

To illustrate, suppose one of the applications, say Application A, deployed on the cluster requires a higher guarantee than other applications that the most up-to-date session state is available. In this case, follow these steps:

- 1. Set the persistence frequency through instance-level settings to time-based (meaning that session state is saved at a regular, configurable time interval) through the Administration interface or the command-line interface.

The persistence frequency for all applications deployed on the cluster is now set to time-based.



2. Change the persistence frequency for Application A to `web-method` (meaning that the session state is saved at the end of every web request, and therefore, there is the highest guarantee that the most up-to-date session information is available) through the `sun-web.xml` file.

The persistence frequency for Application A is changed to `web-method`. The persistence frequency for all other applications remains set to `time-based`.

For more information on persistence frequency, see [“Setting the Persistence Frequency” on page 492](#).

---

<b>NOTE</b>	Session state is not consistently failed over in a cluster unless you enable availability and set the persistence type to <code>ha</code> for every instance in the cluster. For more information on enabling availability for application server instances, see <a href="#">“Enabling and Disabling Availability” on page 474</a> . For more information on setting the persistence type, see <a href="#">“Setting the Persistence Type” on page 485</a> .
-------------	---

---

---

<b>NOTE</b>	If you make session persistence configuration changes for an application, you must redeploy the application before the changes can take effect.
-------------	---

---



# Administering the High-Availability Database (Enterprise Edition)

This chapter describes the high-availability database (HADB) in the Sun Java™ System Application Server Enterprise Edition 7 environment and explains how to configure and administer the HADB. The tasks described in this chapter fit into the overall steps to set up the HADB, which are summarized in the following table.

**Table 21-1** HADB Roadmap

Step	Description of Task	Location of Instructions
1	Decide on your high-availability topology and set up your systems	<i>Sun Java System Application Server System Deployment Guide</i>
2	Install the HADB software with or without the Sun Java System Application Server software	<i>Sun Java System Application Server Installation Guide</i>
3	Set up the following for HADB machines: <ul style="list-style-type: none"> <li>• Shared memory</li> <li>• Communication using <code>rsh</code> or <code>ssh</code></li> <li>• Environment variables</li> </ul>	<i>Sun Java System Application Server Installation Guide</i>
4	Create and start the HADB* Administer the HADB	Current chapter
5	Set up session persistence and the session persistence store*	<a href="#">Chapter 20, “Configuring Session Persistence (Enterprise Edition)”</a>
6	Tune the HADB for maximum performance	<i>Sun Java System Application Server Performance Tuning Guide</i>

\* Can be done as part of cluster setup using the `clsetup` command. See the *Sun Java System Application Server Installation Guide*.

---

**NOTE** If you have not configured `rsh` or `ssh`, the `hadbm` commands described in this chapter will not work. For details about `rsh` or `ssh` configuration, see the *Sun Java System Application Server Installation Guide* and “Setting the Management Protocol” on page 514.

---

This chapter contains the following sections:

- [About the High-Availability Database](#)
- [Configuring the HADB](#)
- [Managing the HADB](#)
- [Expanding the HADB](#)
- [Monitoring the HADB](#)
- [Maintaining the HADB Machines](#)
- [Viewing and Modifying Configuration Attributes](#)
- [Clearing and Archiving History Files](#)
- [Recovering from Session Data Corruption](#)
- [Using Sun Customer Support for the HADB](#)
- [Environment Variables](#)

## About the High-Availability Database

This section introduces you to the high-availability database (HADB), which you can use to store persistent HTTP session and stateful session bean (SFSB) session information. This section includes these topics:

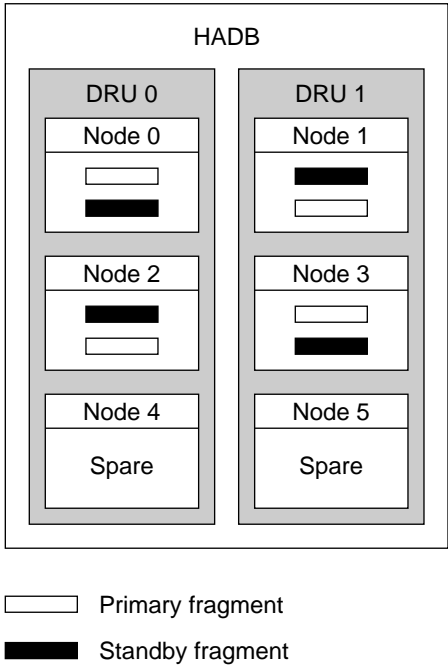
- [HADB Architecture](#)
- [HADB Nodes](#)
- [The `hadbm` Command](#)
- [Creating Other Management Clients](#)

# HADB Architecture

*High-availability* means availability despite planned outages for upgrades or unplanned outages caused by hardware or software failures. The HADB is based on a simple data model and the Always-On technology. The HADB offers an ideal platform for delivering all types of session state persistence within a high performance enterprise application server environment.

The following figure shows the architecture of a database with four active nodes and two spare nodes. Nodes 0 and 1 are a mirror node pair, as are nodes 2 and 3.

**Figure 21-1** HADB Architecture



The HADB achieves high 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. This hash function 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.

HADB nodes are organized into two Data Redundancy Units (DRUs), which mirror each other. Each DRU consists of half of the active and spare nodes, and contains one complete copy of the data. To ensure fault tolerance, the computers that support one DRU must be completely self-supported with respect to power (use of uninterruptible power supplies is recommended), processing units, and storage. If a power failure occurs in one DRU, the nodes in the other DRU can continue servicing requests until the power returns.

Without a session persistence mechanism, the HTTP or SFSB session state, including the passivated session state, is lost when one web or EJB container fails over to another. Use of the HADB for session persistence overcomes this situation. The HADB stores and retrieves state information in a separate but well-integrated persistent storage tier.

The HADB reclaims space when session data is deleted. The HADB places session data records in fixed size blocks. When all records of a block are deleted, the block is freed. Records of a block can be deleted randomly, creating “holes” in the block. When a new record is to be inserted into a block and contiguous space is needed, the holes are removed and thus the block is compacted.

This is a brief summary of the architecture. For details, see the *Sun Java System Application Server System Deployment Guide*.

## HADB Nodes

A database node consists of a set of processes, a dedicated area of shared memory, and one or more secondary storage devices. It is used for storing and updating session data. Each node must have a mirror node, therefore nodes occur in pairs. In addition, to maximize availability, you should include two or more spare nodes, one in each DRU, so if a node fails a spare can take over while the node is repaired.

For an explanation of node topology alternatives, see the *Sun Java System Application Server System Deployment Guide*. For more general information about nodes and how to monitor them, see [“Node Status” on page 540](#).

## The hadbm Command

---

**NOTE** If you have not configured `rsh` or `ssh`, the `hadbm` command will not work. For details about `rsh` or `ssh` configuration, see the *Sun Java System Application Server Installation Guide* and [“Setting the Management Protocol” on page 514](#).

---

The `hadbm` command is the management client for the HADB database. By default, there is one management client located in the `install_dir/SUNWhadb/4/bin` directory. For information about creating additional management clients, see [“Creating Other Management Clients” on page 512](#).

The commands you use to administer HADB are subcommands of the `hadbm` command. The general syntax is as follows:

```
hadbm subcommand [-short-form [argument]]* [--long-form [argument]]* [dbname]
```

For example, the following is one use of the `hadbm status` subcommand, which lets you check the status of HADB:

```
hadbm status --nodes
```

The subcommand identifies the operation or task you wish to perform. Subcommands are case-sensitive.

Options are also case-sensitive. Each option has a long form and a short form. Short forms have a single dash (-); while long forms have two dashes (--). Options modify how `hadbm` performs a subcommand. Most options require argument values, except for boolean options, which must be present to switch a feature on. Optional options are enclosed in square brackets [ ] in syntax lines.

For subcommands that take a database name, if a database is not identified, the default database is used. The default database is `hadb`, all lowercase.

You can set the password for a subcommand from a file instead of entering the password at the command line. The `--dbpasswordfile` option takes the file containing the passwords. The valid contents for the file are:

```
HADB_DBPASSWORD=password
```

The rest of the contents of the file are ignored. If both the `--dbpassword` and `--dbpasswordfile` options are specified, the `--dbpassword` takes precedence. If a password is required, but is not specified in the command, you are prompted for a password.

The `hadbm` general command options, which you can use with any `hadbm` subcommand, are listed in the following table. All are booleans that are not present by default.

**Table 21-2** `hadbm` General Options

Long Form	Short Form	Description
<code>--quiet</code>	<code>-q</code>	Executes the subcommand silently without any descriptive messages.

**Table 21-2** `hadbm` General Options (*Continued*)

Long Form	Short Form	Description
<code>--help</code>	<code>-?</code>	Displays a brief description of this command and all the supported subcommands. No subcommand is required.
<code>--version</code>	<code>-V</code>	Displays the version details of the <code>hadbm</code> command. No subcommand is required.
<code>--yes</code>	<code>-y</code>	Executes the subcommand in non-interactive mode.
<code>--force</code>	<code>-f</code>	Executes the command non-interactively and does not throw an error if the command's post condition is already achieved.
<code>--echo</code>	<code>-e</code>	Displays the subcommand with all the options and their user-defined values or the default values, then executes the subcommand.

## Creating Other Management Clients

By default, the `hadbm` command must be executed from the machine from which `hadbm create` was executed, or from an HADB machine, by the user who executed `hadbm create` (the HADB system user). There are some scenarios for which this limitation is not desirable. It may be useful to give more than one user access to the database. The machine from which `hadbm create` was executed may be unavailable due to a hardware failure.

When the HADB system user executes `hadbm create`, the name of the database and the location of its configuration files are recorded by the management client in a file named `.cladmrc` in the HADB system user's home directory.

To identify a database, all `hadbm` commands except `create`, `list`, `help`, and `version search` for the database configuration files in these locations, in the following order:

- The location listed in the `.cladmrc` file of the HADB system user's home directory
- The default path, `/etc/opt/SUNWhadb/dbdef`
- `installpath/etc/dbdef` (*installpath* is `install_dir/SUNWhadb/4/` by default)

If the configuration files cannot be located, `hadbm` cannot execute properly. The `hadbm list` command does not list the database, and you receive the following error:

```
hadbm:Error 22002: Specified database does not exist. Use "hadbm list" to
get a list of existing databases: [ hadb ]
```



You can set up an `hadbm` management client on a non-HADB machine other than the machine from which `hadbm create` was executed by ensuring that the new management client can find the database configuration files.

To set up a new `hadbm` management client:

1. Make sure that the HADB software is installed on the new management client machine. See the *Sun Java System Application Server Installation Guide*.
2. Copy the `.cladmrc` file from the HADB system user's home directory to the home directory of the user of the new management client machine.
3. Check whether the configuration files are accessible from the new management client machine. Entries in the `.cladmrc` file have the following format:

```
dbname: configpath: howtoaccess
```

The *howtoaccess* value can be `NFSMNT` for the network file system, or a host name for a path that can only be accessed locally. For example:

```
hadb: /etc/opt/SUNWhadb/dbdef:NFSMNT
hadb: /dsk0/dbdef:host0
```

If the *howtoaccess* value of the database is `NFSMNT` and the *configpath/dbname/cfg* file can be read from the new management client machine, [Step 4](#) is unnecessary.

4. If the configuration files are not accessible from the new management client machine:
  - a. Create a directory on the new management client machine with the same path as the *configpath*. For example, this new directory could be `/etc/opt/SUNWhadb/dbdef`.
  - b. Create a directory called *dbname* (for example `hadb`) under *configpath*. Copy into this new directory all the files from *configpath/dbname* on an HADB machine or the machine from which `hadbm create` was executed.
  - c. Change the *howtoaccess* value of the database's entry in the `.cladmrc` file to the host name of the new management client machine.
5. Make sure the new management client machine can communicate with the HADB machines over the network through the ManagementProtocol (`ssh` or `rsh`) that was specified in the `hadbm create` command.

# Configuring the HADB

This section describes the following basic HADB configuration tasks:

- [Setting the Management Protocol](#)
- [Creating a Database](#)
- [Setting Heterogeneous Device Paths](#)
- [Configuring Double Networks](#)
- [Additional Steps for inetd](#)
- [Setting Up the JDBC Connection Pool](#)

## Setting the Management Protocol

Before you create the database, you need to determine whether the `hadbm` command uses `rsh` or `ssh` for remote execution. Because `ssh` is the default, you only need to set the management protocol if you need to use `rsh` instead. Use `--set` option of the `hadbm create` command as in the following example:

```
hadbm create --set ManagementProtocol=rsh --spares 2 --devicesize 1024
--dbpassword secret123 --hosts n0,n1,n2,n3,n4,n5
```

---

<b>NOTE</b>	Using <code>ssh</code> is strongly recommended because <code>ssh</code> is more secure than <code>rsh</code> .
-------------	--

---

The `hadbm` command uses the `rsh` or `ssh` protocol to communicate with the database nodes. The management protocol is not used by HADB nodes for communication.

## Creating a Database

The `clsetup` command creates an HADB database as part of cluster initialization and setup. This is the recommended way of creating a database. For details about `clsetup`, see the *Sun Java System Application Server Installation Guide*.

However, you can create a database outside of `clsetup`. To manually create a database to store the session data, use the `hadbm create` command. The syntax is as follows.

```
hadbm create [--installpath=path] [--historypath=path] [--devicepath=path]
[--configpath=path] [--datadevices=devices-per-node] [--portbase=base-no]
[--spares=sparecount] [--set=attr-name-value-list] [--inetd] [--inetdsetupdir=path]
--devicesize=size --dbpassword=password | --dbpasswordfile=file --hosts=node-list
[dbname]
```

For example:

```
hadbm create --spares 2 --devicesize 1024 --dbpassword secret123 --hosts
n0,n1,n2,n3,n4,n5
```

The `hadbm create` command also starts the database if the `--inetd` and `--inetdsetupdir` options are not used.

If you have difficulty creating a database, check the following:

- File and directory permissions must be set to allow read, write, and execute access to the install, history, device, and config paths for the following users:
  - Sun Java System Application Server administrative user (set during installation)
  - HADB system user

For details about setting user permissions, see the *Sun Java System Application Server Installation Guide*.

- Sun Java System Application Server and HADB port assignments must not conflict with other port assignments on the same machine. Default and recommended port assignments are as follows:
  - Sun Java System Message Queue: 7676
  - IIOP: 3700
  - HTTP server (UNIX root or Windows): 80
  - HTTP server (UNIX non-root): 1024
  - Admin server (UNIX root or Windows): 4848

- HADB nodes: Each node uses six consecutive ports. If the default portbase (15200) is used, node 0 uses 15200 through 15205, node 1 uses 15220 through 15225, and so on.
- Disk space must be adequate; see the *Sun Java System Application Server Installation Guide*.

Database creation errors are written to the following files:

- The Sun Java System Application Server log file. See [Chapter 5, “Using Logging.”](#)
- The HADB history file. See [“Clearing and Archiving History Files” on page 553.](#)
- The operating system’s syslog files if the SysLogging configuration attribute is set to TRUE, the default. See [“Configuration Attributes” on page 548.](#)

**NOTE**

If for any reason the `hadbm create` command is not successfully executed, history files on the HADB machines are deleted.

However, if the management-client machine from which the `hadbm create` command is run is different from the HADB machines but has a directory with the same path as the `historypath` on the HADB machines, a copy of the history files is saved in this directory on the management-client machine. These history files can be used to analyze why the `hadbm create` command was not successfully executed.

If you still have difficulty creating a database, contact Sun customer support. See [“Using Sun Customer Support for the HADB” on page 554.](#)

The `hadbm create` command options are listed in the following table.

**Table 21-3** `hadbm create` Options

Long Form	Short Form	Default	Description
<code>--installpath</code>	<code>-n</code>	parent of the directory where <code>hadbm</code> resides: <code>install_dir/SUNWhadb/4/</code>	Specifies the HADB system installation path. This path must already exist and be writable. Use this option if the HADB server installation resides in a location different from the management-client machine from which the <code>hadbm create</code> command is run.

**Table 21-3** `hadbm create` Options (Continued)

Long Form	Short Form	Default	Description
<code>--historypath</code>	<code>-t</code>	<code>/var/tmp</code>	<p>Specifies the path to the history files. This path must already exist and be writable. For details about history files, see <a href="#">“Clearing and Archiving History Files” on page 553</a>.</p> <p>When database creation fails, the history files are removed from the HADB machines, and valuable debugging information is lost. However, if you create a writable directory with the same path as the <code>--historypath</code> on the machine from which you execute the <code>hadbm create</code> command, and this machine is different from the HADB machines, the history files are saved there.</p>
<code>--devicepath</code>	<code>-d</code>	<code>/var/opt/SUNWhadb</code>	<p>Specifies the path to the devices. There are three devices: the <code>DataDevice</code>, the <code>NiLogDevice</code> (node internal log device), and the <code>RelalgDevice</code> (relational algebra query device). This path must already exist and be writable. To set this path differently for each node or each device, see <a href="#">“Setting Heterogeneous Device Paths” on page 519</a>.</p>
<code>--configpath</code>	<code>-c</code>	<code>/etc/opt/SUNWhadb</code>	<p>Specifies the path to the configuration files used internally by the HADB. This path must already exist and be writable.</p>
<code>--datadevices</code>	<code>-a</code>	<code>1</code>	<p>Specifies the number of data devices on each node, between 1 and 8 inclusive. Data devices are numbered starting at 0.</p>
<code>--portbase</code>	<code>-b</code>	<code>15200</code>	<p>Specifies the port base number used for node 0. Successive nodes are automatically assigned port base numbers in steps of 20 from this number. Each node uses its port base number and the next five consecutively numbered ports.</p> <p>If you want to run several databases on the same machine, you should have a plan for allocating port numbers and allocate them explicitly.</p>
<code>--spares</code>	<code>-s</code>	<code>0</code>	<p>Specifies the number of spare nodes. This number must be even and must be less than the number of nodes specified in the <code>--hosts</code> option. Spare nodes are optional, but having two or more ensures high availability.</p>

**Table 21-3** `hadbm create` Options (Continued)

Long Form	Short Form	Default	Description
<code>--set</code>	<code>-S</code>	none	<p>Specifies a comma-separated list of database configuration attributes in <i>name=value</i> format. For explanations of valid database configuration attributes, see <a href="#">“Viewing and Modifying Configuration Attributes” on page 546</a>.</p> <p>For example, to specify the use of <code>rsh</code> instead of <code>ssh</code> (the default), use the following option:</p> <pre>--set ManagementProtocol=rsh</pre> <p>To use <code>--set</code> to set the <code>--devicepath</code> differently for each node or each device, see <a href="#">“Setting Heterogeneous Device Paths” on page 519</a>.</p>
<code>--inetd</code>	<code>-I</code>	not specified	<p>If specified, the database is configured to run with the <code>inetd</code> daemon, and is <i>not</i> automatically started after it is created. See <a href="#">“Additional Steps for inetd” on page 520</a>.</p>
<code>--inetdsetupdir</code>	<code>-u</code>	current directory	<p>Specifies the directory in which to store the <code>inetd</code> setup files. The directory must exist and be writable.</p>
<code>--devicesize</code>	<code>-z</code>	none	<p>Specifies the size of each device in MB. The device size should be as large as possible. The recommended size is four times the expected size of the user data, based on the number of users and the size of each user record.</p> <p>The maximum size is the maximum operating system file size or 256 GB, whichever is smaller. The minimum size is as follows:</p> <pre>(4 x LogbufferSize + 16MB) / --datadevices</pre> <p>You can increase the device size later as described in <a href="#">“Adding Storage Space to Existing Nodes” on page 533</a>.</p> <p>For more information on setting the <code>LogbufferSize</code>, see <a href="#">“Viewing and Modifying Configuration Attributes” on page 546</a>.</p>
<code>--dbpassword</code>	<code>-p</code>	none	<p>Creates a password for the HADB system user. Must be at least 8 characters. You can use <code>--dbpasswordfile</code> instead. For details, see <a href="#">“The hadbm Command” on page 510</a>.</p>
<code>--dbpasswordfile</code>	<code>-P</code>	none	<p>Specifies a file that stores the password to be created for the HADB system user. For details, see <a href="#">“The hadbm Command” on page 510</a>.</p>

**Table 21-3** `hadbm create` Options (*Continued*)

Long Form	Short Form	Default	Description
<code>--hosts</code>	<code>-H</code>	<code>none</code>	<p>Specifies a comma-separated list of host names or IP addresses for the nodes in the database. Using IP addresses is recommended because there is no dependence on DNS lookups. Host names must be absolute. You cannot use <code>localhost</code> or <code>127.0.0.1</code> as a host name.</p> <p>One node is created for each comma-separated item in the list. The number of nodes must be even. Using duplicate host names creates multiple nodes on the same machine with different port numbers. Make sure that nodes on the same machine are not mirror nodes.</p> <p>Nodes are numbered starting at 0 in the order listed in this option. The first mirrored pair are nodes 0 and 1, the second 2 and 3, and so on. Odd numbered nodes are in one DRU, even numbered nodes in the other. If <code>--spares</code> is used, spare nodes are those with the highest numbers.</p> <p>For information about configuring double network interfaces, see <a href="#">“Configuring Double Networks” on page 520</a>.</p>
<code>dbname</code>	<code>none</code>	<code>hadb</code>	<p>Specifies the database name, which must be unique. To make sure the database name is unique, use the <code>hadbm list</code> command to list existing database names.</p> <p>Use the default database name unless you need to create multiple databases. For example, to create multiple clusters with independent databases on the same set of HADB machines, use a separate database name for each cluster.</p>

## Setting Heterogeneous Device Paths

You can use the `--set` option of `hadbm create` to set the `--devicepath` differently for each node or each device. There are three types of devices: the `DataDevice`, the `NiLogDevice` (node internal log device), and the `RelalgDevice` (relational algebra query device). The syntax for each *name=value* pair is as follows, where *-devno* is required only if the *device* is `DataDevice`:

`Node=nodeno.device-devno.Path=path`

For example:

```
--set Node-0.DataDevice-0.Path=/disk0,Node-1.DataDevice-0.Path=/disk1
```

Any device path that is not set for a particular node or device defaults to the `--devicepath` value. You cannot change device paths using the `hadbm set` or `hadbm addnodes` commands.

## Configuring Double Networks

To allow the HADB to tolerate single network failures, you can equip each HADB machine with two NIC cards. For each machine, the IP addresses of each of the NIC cards must be on separate IP subnets.

During database creation, you specify two IP addresses or host names for each node, one for each NIC card IP address, using the `--hosts` option. For each node, the first IP address is on `net-0` and the second on `net-1`. The syntax is as follows, with host names for the same node separated by a plus sign (+):

```
--hosts=node0net0name+node0net1name,node1net0name+node1net1name,node2net0name+node2net1name, ...
```

For example, the following creates two nodes, each with two host names. The host names for node 0 are `n0a` and `n0b`, and the host names for node 1 are `n1a` and `n1b`. The `n0a` and `n1a` hosts are on `net-0`, and the `n0b` and `n1b` hosts are on `net-1`.

```
--hosts n0a+n0b,n1a+n1b
```

Within a database, all nodes must have one host name, or all nodes must have two host names. A database cannot have a mixture of nodes with double host names and single host names.

## Additional Steps for inetd

If you did not specify `--inetd` when you created the database, the database is initialized and started from the `--installpath` location, which you can verify using the `hadbm status` command.

If you specified `--inetd`, the database exists, as shown by the `hadbm list` command, but is not running, as shown by `hadbm status` command.

Using `inetd` in production environments is recommended, but it is usually not needed in development or test environments. Creating a database using the `--inetd` option allows you to use `inetd` to automatically restart HADB nodes on a machine if the machine reboots. This allows for a more robust deployment, but it has its drawbacks in terms of administration.



In particular, if you want to stop a node, for example to perform some kind of maintenance, you must perform the following tasks:

1. Remove the `inetd` entry for that node from the `inetd` configuration files, or the node is automatically restarted as soon as you stop it. Then reconfigure `inetd` as follows:

```
kill -HUP inetd-process-id
```

2. Re-add the entry to the `inetd` files once you have restarted the node, then reconfigure `inetd`.

Also, if you add nodes, you must take extra steps to update the `inetd` configuration files to take into account the new nodes.

To set up `inetd` for a new database or an existing database with new nodes, follow these steps:

1. Stub files for `inetd` support are created in the directory specified in the `--inetdsetupdir` option and are named as follows:

```
dbname.hostname.inetd.conf
```

```
dbname.hostname.services
```

All stub files are placed locally on the machine where the `hadbm create` command is typed. Host names are further distinguished by numbers appended to the *hostname*.

Append the contents of these files to the `/etc/inetd.conf` and `/etc/services` files for each node's machine, then reconfigure `inetd` as follows:

```
kill -HUP inetd-process-id
```

2. After `inetd` has been reconfigured on each machine, you are ready to initialize and start your database using the `hadbm clear` command. See [“Clearing the HADB” on page 531](#). This step is necessary for a new database, but not for an existing database with new nodes.
3. You can verify that `inetd` is working by stopping a node and checking to make sure it restarts. See [“Stopping a Node” on page 527](#) and [“Getting the Status of the HADB” on page 539](#).

## Setting Up the JDBC Connection Pool

The Sun Java System Application Server communicates with the HADB in the same way that it communicates with relational databases used for data storage, therefore you need to set up a JDBC connection pool for the HADB as you would for any other database.

Using the `clsetup` command is recommended for configuring a JDBC connection pool and JDBC resource for the HADB in the Sun Java System Application Server. This command is described in the *Sun Java System Application Server Installation Guide*.

Manual configuration of a JDBC connection pool and JDBC resource for the HADB is briefly summarized in these sections:

- [Getting the JDBC URL](#)
- [Creating a Connection Pool](#)
- [Creating a JDBC Resource](#)

For general information about connection pools and JDBC resources, see [“About JDBC Resources” on page 241](#).

### Getting the JDBC URL

Before you can set up the JDBC connection pool, you need to determine the JDBC URL of the HADB using the `hadbm get` command as follows:

```
hadbm get JdbcUrl [dbname]
```

For example:

```
hadbm get JdbcUrl
```

The JDBC URL is displayed on the standard output device in the following form:

```
jdbc:sun:hadb:host:port,host:port,...
```

Remove the `jdbc:sun:hadb:` prefix and use the `host:port,host:port...` part as the value of the `serverList` connection pool property, described in the next section.

## Creating a Connection Pool

The following table summarizes connection pool settings required for the HADB. Change Steady Pool Size when adding nodes, but do not change other settings.

**Table 21-4** HADB Connection Pool Settings

Setting	Required Value for HADB
Name	The HADB JDBC resource's Pool Name setting must refer to this name
Database Vendor	HADB 4.3
Global Transaction Support	Unchecked/false
DataSource Classname	<code>com.sun.hadb.jdbc.ds.HadbDataSource</code>
Steady Pool Size	Use 8 connections for each active HADB node. For more detailed information, see the <i>System Deployment Guide</i> .
Connection Validation Required	Checked/true
Validation Method	<code>meta-data</code>
Table Name	Do not specify
Fail All Connections	Unchecked/false
Transaction Isolation	<code>repeatable-read</code>
Guarantee Isolation Level	Checked/true

The following table summarizes connection pool properties required for the HADB. Change `serverList` when adding nodes, but do not change other properties.

**Table 21-5** HADB Connection Pool Properties

Property	Description
<code>username</code>	Specifies the name of the <code>storeuser</code> to be specified in the <code>asadmin create-session-store</code> command. See <a href="#">“Creating the Session Store” on page 468</a> .
<code>password</code>	Specifies the <code>storepassword</code> to be specified in the <code>asadmin create-session-store</code> command. See <a href="#">“Creating the Session Store” on page 468</a> .

**Table 21-5** HADB Connection Pool Properties (*Continued*)

Property	Description
serverList	<p>Specifies the JDBC URL of the HADB. To determine this value, see <a href="#">“Getting the JDBC URL” on page 522</a>.</p> <p>You must change this value if you add nodes to the database. See <a href="#">“Adding Nodes to the HADB” on page 534</a>.</p>
cacheDatabaseMetaData	<p>Setting this property to <code>false</code> as required ensures that calls to <code>Connection.getMetaData()</code> make calls to the database, which ensures that the connection is valid.</p>
eliminateRedundantEndTransaction	<p>Setting this property to <code>true</code> as required improves performance by eliminating redundant commit and rollback requests and ignoring these requests if no transaction is open.</p>
maxStatement	<p>Specifies the maximum number of statements per open connection that are cached in the driver statement pool. Set this property to 20.</p>

### Connection Pool Example

Here is an example `asadmin create-jdbc-connection-pool` command that creates an HADB JDBC connection pool. For more details about this command, see the *Sun Java System Application Server Developer’s Guide to J2EE Services and APIs*.

```
asadmin create-jdbc-connection-pool --user adminname --password secret
--datasourceclassname com.sun.hadb.jdbc.ds.HadbDataSource
--steadypoolsize=32 --isolationlevel=repeatable-read
--isconnectvalidatereq=true --validationmethod=meta-data --property
username=storename:password=secret456:serverList=host\\:port,host\\:port,host\\:p
ort,host\\:port,host\\:port,host\\:port:cacheDatabaseMetaData=false:eliminateRedund
antEndTransaction=true hadbpool
```

Note that colon characters (:) within property values must be escaped with double backslashes (\\) on Solaris™ platforms, because otherwise they are interpreted as property delimiters. On Windows platforms, colon characters (:) must be escaped with single backslashes (\). For details about using escape characters, see [“Using Escape Characters” on page 570](#).

## Creating a JDBC Resource

The following table summarizes JDBC resource settings required for the HADB.

**Table 21-6** HADB JDBC Resource Settings

Setting	Description
JNDI Name	<p>The following JNDI name is the default in the session persistence configuration: <code>jdbc/hastore</code>. You can use the default name or a different name.</p> <p>You must also specify this JNDI name as the value of the <code>store-pool-jndi-name</code> Persistence Store property when you activate the availability service. See <a href="#">“Referencing the HADB Database’s JDBC Resource” on page 467</a>.</p>
Pool Name	Select from the list the name (or ID) of the HADB connection pool used by this JDBC resource. For more information, see <a href="#">“Creating a Connection Pool” on page 523</a> .
Data Source Enabled	Checked/true

## Managing the HADB

In general, management operations are not necessary unless you are replacing or upgrading your network, hardware, operating system, or HADB software. For assistance with these operations, call Sun customer support. See [“Using Sun Customer Support for the HADB” on page 554](#). The following sections explain various management operations:

- [Starting a Node](#)
- [Stopping a Node](#)
- [Restarting a Node](#)
- [Starting the HADB](#)
- [Stopping the HADB](#)
- [Restarting the HADB](#)
- [Listing Databases](#)
- [Clearing the HADB](#)
- [Removing a Database](#)

# Starting a Node

If no node is running in the database, use `hadbm clear` to start the nodes even if you are running with `inetd`. See [“Clearing the HADB” on page 531](#).

You may want to start a node in the following circumstances:

- If you have stopped a node, for example for hardware or software replacement. See [“Stopping a Node” on page 527](#).
- If a node has stopped due to a hardware failure, after the hardware has been mended.
- If a node has stopped due to a software failure and the node was unable to recover automatically.

If a node has stopped due to a failure, starting a node is not necessary if you are running with `inetd`, because after the machine reboots, `inetd` automatically restarts the node.

In most cases, you should first attempt to start the node using the `normal` start level. You must use the `repair` start level if starting a node using the `normal` start level fails or times out.

To start a node in the database, use the `hadbm startnode` command. The syntax is as follows:

```
hadbm startnode [--startlevel=level] nodeno [dbname]
```

For example:

```
hadbm startnode 1
```

The `hadbm startnode` command options are listed in the following table.

**Table 21-7** `hadbm startnode` Options

Long Form	Short Form	Default	Description
<code>--startlevel</code>	<code>-l</code>	<code>normal</code>	Specifies the level at which you want to start the node. Valid levels are <code>normal</code> and <code>repair</code> . Starting a node at the <code>repair</code> level forces an active node to repair data from its mirror node.
<code><i>nodeno</i></code>	<code>none</code>	<code>none</code>	Specifies the node you want to start. You can use the <code>hadbm status</code> command to display the numbers of all nodes in a database.
<code><i>dbname</i></code>	<code>none</code>	<code>hadb</code>	Specifies the database name.

## Stopping a Node

You may want to stop a node if you want to replace hardware or software on the machine, and you need to stop the machine.

---

**CAUTION** Do not stop a node if its mirror node is not running, because this may force the database into a Non Operational state. For details about node status, see [“Getting the Status of the HADB” on page 539](#).

---

To stop a node in the database, use the `hadbm stopnode` command. The syntax is as follows:

```
hadbm stopnode [--no-repair] nodeno [dbname]
```

For example:

```
hadbm stopnode 1
```

---

**NOTE** If you have set up `inetd`, the node supervisor is automatically restarted when you stop a node. The result is that the node resumes the running state but is in the offline role. A node that is in the running state and has the offline role is equivalent to a stopped node. A host machine in which all nodes have offline roles can be shut down safely.

If you stop a node with `hadbm stopnode`, you must start it with `hadbm startnode`, even if `inetd` is used, because `inetd` can't start an offline node.

---

The `hadbm stopnode` command options are listed in the following table.

**Table 21-8** `hadbm stopnode` Options

Long Form	Short Form	Default	Description
<code>--no-repair</code>	<code>-R</code>	not present	If present, the node is stopped and no spare node replaces the stopped node. By default, a spare node starts up and takes over the functioning of the stopped node.

---

**Table 21-8** `hadbm stopnode` Options (*Continued*)

Long Form	Short Form	Default	Description
<i>nodeno</i>	none	none	Specifies the node you want to stop. You can use the <code>hadbm status</code> command to display the numbers of all nodes in a database. The mirror node of this node number must be running.
<i>dbname</i>	none	hadb	Specifies the database name.

## Restarting a Node

You may want to restart a node if you notice strange behavior in a node (for example excessive CPU consumption) and want to check whether a restart cures the problem.

**CAUTION** Do not stop a node if its mirror node is not running, because this may force the database into a Non Operational state. For details about node status, see [“Getting the Status of the HADB” on page 539](#).

To restart a node in the database, use the `hadbm restartnode` command. The syntax is as follows:

```
hadbm restartnode [--startlevel=level] nodeno [dbname]
```

For example:

```
hadbm restartnode 1
```

The `hadbm restartnode` command options are listed in the following table.

**Table 21-9** `hadbm restartnode` Options

Long Form	Short Form	Default	Description
<code>--startlevel</code>	<code>-l</code>	<code>normal</code>	Specifies the level at which you want to start the node. Valid levels are <code>normal</code> and <code>repair</code> . Using the <code>normal</code> start level when restarting a node is generally recommended. Starting a node at the <code>repair</code> level forces an active node to repair data from its mirror node.



**Table 21-9** `hadbm restartnode` Options (*Continued*)

Long Form	Short Form	Default	Description
<i>nodeno</i>	none	none	Specifies the node you want to start. You can use the <code>hadbm status</code> command to display the numbers of all nodes in a database.
<i>dbname</i>	none	hadb	Specifies the database name.

## Starting the HADB

To start a database, use the `hadbm start` command. The syntax is as follows:

```
hadbm start [dbname]
```

For example:

```
hadbm start
```

The default *dbname* is `hadb`, all lowercase.

This command starts all nodes that were running before the database was stopped. Individually stopped (offline) nodes are not started when the database is started after a stop.

## Stopping the HADB

When you stop and start the HADB in separate operations, data is unavailable while the HADB is stopped. To keep data available, you can restart the HADB as described in [“Restarting the HADB” on page 530](#).

You may want to stop the HADB in the following circumstances:

- If you want to remove the HADB database.
- If you want to perform system maintenance that affects all HADB nodes.
- Before executing the `hadbm clear` command to reinitialize the database. See [“Clearing the HADB” on page 531](#).

Before stopping the HADB, you should either stop dependent Sun Java System Application Server instances or configure them to use a different persistence method. For details, see [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\).”](#)

---

**NOTE** If you stop the HADB with `hadbm stop`, you must start it with `hadbm start`, even if `inetd` is used, because `inetd` can't start offline nodes.

---

To stop a database, use the `hadbm stop` command. The syntax is as follows:

```
hadbm stop [dbname]
```

For example:

```
hadbm stop
```

The default *dbname* is `hadb`, all lowercase. For more information about database states, see [“Getting the Status of the HADB” on page 539](#).

When you stop the database, all the running nodes in the database are stopped and the status of the database is Stopped.

If you have set up `inetd` to automatically restart the HADB, you must perform these steps to stop the HADB:

1. Edit the `/etc/inetd.conf` and `/etc/services` files to remove the lines you added when you created the database.
2. Reconfigure `inetd` as follows:

```
kill -HUP inetd-process-id
```
3. Use the `hadbm stop` command.

## Restarting the HADB

You may want to restart the HADB if you notice strange behavior in the HADB (for example consistent timeout problems) and want to check whether a restart cures the problem.

When you restart the HADB, data and database services remain available. When you stop and start the HADB in separate operations, data and database services are unavailable while the HADB is stopped. This is because `hadbm restart` performs a rolling restart of nodes: it stops and starts the nodes one by one. In contrast, `hadbm stop` stops all nodes simultaneously.

If an `hadbm set` command fails, restarting the HADB restores the previous configuration. For details about `hadbm set`, see [“Viewing and Modifying Configuration Attributes” on page 546](#).

To restart a database, use the `hadbm restart` command. The syntax is as follows:

```
hadbm restart [--no-rolling] [dbname]
```

For example:

```
hadbm restart
```

The default *dbname* is `hadb`, all lowercase. By default, this command restarts each of the nodes in the database to the current state or a better state. If you specify the `--no-rolling` or `-g` option, this command restarts all nodes at once, with loss of service.

## Listing Databases

To list all the databases that have been created, use the `hadbm list` command. The syntax is as follows:

```
hadbm list
```

## Clearing the HADB

You may want to clear the HADB in the following circumstances:

- If you are creating a database that uses `inetd`. See [“Additional Steps for inetd” on page 520](#).
- If the `hadbm status` command reveals that the database is Non Operational or that multiple nodes are in the Waiting state. See [“Getting the Status of the HADB” on page 539](#).
- If you are recovering from session data corruption. See [“Recovering from Session Data Corruption” on page 554](#).

The `hadbm clear` command stops the database nodes, clears the database devices, then starts the nodes. The syntax is as follows.

```
hadbm clear [--fast] [--spares=sparecount] --dbpassword=password |  
--dbpasswordfile=file [dbname]
```

For example:

```
hadbm clear --fast --spares=2 --dbpassword secret123
```

The `hadbm clear` command options are listed in the following table.

**Table 21-10** `hadbm clear` Options

Long Form	Short Form	Default	Description
<code>--fast</code>	<code>-F</code>	not present	If present, skips device initialization while initializing the database. Do not use if the disk storage device is corrupted or if you have just created the database and set up <code>inetd</code> .
<code>--spares</code>	<code>-s</code>	previous number of spares	Specifies the number of spare nodes the reinitialized database will have. This number must be even and must be less than the number of nodes in the database. Spare nodes are optional, but having two or more ensures high availability.
<code>--dbpassword</code>	<code>-p</code>	none	Specifies the HADB system user password. You can use <code>--dbpasswordfile</code> instead. For details, see <a href="#">“The hadbm Command” on page 510</a> .
<code>--dbpasswordfile</code>	<code>-P</code>	none	Specifies a file that stores the password for the HADB system user. For details, see <a href="#">“The hadbm Command” on page 510</a> .
<i>dbname</i>	none	hadb	Specifies the database name.

## Removing a Database

The database you want to remove must exist and must be in the Stopped state. See [“Stopping the HADB” on page 529](#). To remove an existing database from the HADB system, use the `hadbm delete` command. The syntax is as follows:

```
hadbm delete [dbname]
```

For example:

```
hadbm delete
```

The default database name is `hadb`, all lowercase. When you execute this command, the configuration files, device files, log files, and history files of the database are deleted, and shared memory resources are freed.

# Expanding the HADB

If you determine that your system performance is limited because the HADB cannot persist data fast enough, you can expand the HADB to increase throughput without shutting down your Sun Java System Application Server cluster or the HADB. This section describes how you can expand the HADB in the following sections:

- [Adding Storage Space to Existing Nodes](#)
- [Adding Machines](#)
- [Adding Nodes to the HADB](#)
- [Refragmenting the HADB](#)
- [Adding Nodes Without Refragmenting](#)

You should also read [“Maintaining the HADB Machines” on page 544](#).

---

**NOTE** You should expand the HADB at a time when your system is lightly loaded. You cannot expand and refragment the HADB by adding nodes if the disks for the HADB nodes are more than half full, because this additional space is needed for the refragmentation operation.

---

## Adding Storage Space to Existing Nodes

You may want to add storage space to the HADB in the following circumstances:

- If there is unused space on the disks on which the HADB nodes reside, or if you have upgraded these disks.
- If one of the following messages appears:
 

```
4592: No free blocks on data devices
4593: No unreserved blocks on data devices
```
- If the `hadbm deviceinfo` command reports insufficient free size. See [“Getting Device Information” on page 542](#).

You can increase the device size in MB using either of the following `hadbm set` commands:

```
hadbm set DataDeviceSize=size
```

```
hadbm set TotalDataDeviceSizePerNode=size
```

For example:

```
hadbm set DataDeviceSize=1024
```

The `TotalDataDeviceSizePerNode` is equal to the `DataDeviceSize` multiplied by the `NumberOfDataDevices`. Therefore, `TotalDataDeviceSizePerNode` and `DataDeviceSize` are mutually dependent: changing one changes the other.

The `DataDeviceSize` should be as large as possible. The recommended size is four times the expected size of the user data, based on the number of users and the size of each user record.

Changing the `DataDeviceSize` or `TotalDataDeviceSizePerNode` on a database in a `FaultTolerant` or higher state means that the system is upgraded without loss of data, and the database remains in an `Operational` state during the reconfiguration. If you change device size on a system that is not `FaultTolerant` or better, data is lost. For more information about database states, see [“Database Status” on page 540](#).

## Adding Machines

You may want to add machines if the HADB requires more processing or storage capacity. For an explanation of node topology alternatives, see the *Sun Java System Application Server System Deployment Guide*.

To add a new machine on which to run the HADB, install the HADB packages with or without the Sun Java System Application Server as described in the *Sun Java System Application Server Installation Guide*.

## Adding Nodes to the HADB

When you create new nodes and add them to the database, you increase processing and storage capacity. To add nodes, use the `hadbm addnodes` command. The syntax is as follows:

```
hadbm addnodes [--no-refragment] [--spares=sparecount] --dbpassword=password |  
--dbpasswordfile=file [--inetdsetupdir=path] --hosts=node-list [dbname]
```

For example:

```
hadbm addnodes --dbpassword secret123 --hosts n6,n7,n8,n9
```

After you have added nodes, you must perform these additional tasks:

- Update the `serverlist` property in the HADB's JDBC connection pool.
- Adjust the Steady Pool Size of the HADB's JDBC connection pool. Generally, you add 8 more connections for each new node. For more detailed information, see the *Sun Java System Application Server System Deployment Guide*.

For details, see [“Setting Up the JDBC Connection Pool” on page 522](#).

---

**NOTE**

If you created the database using `--inetd`, you must do the following:

- Use the `--no-refragment` option and refragment the database in a separate step using the `hadbm refragment` command.
  - Take extra steps to update the `inetd` configuration files to take into account the new nodes you add. For more information, see [“Additional Steps for inetd” on page 520](#).
- 

The `hadbm addnodes` command options are listed in the following table.

**Table 21-11** `hadbm addnodes` Options

Long Form	Short Form	Default	Description
<code>--no-refragment</code>	<code>-r</code>	not specified	<p>If specified, does not refragment the database during node creation; you can refragment the database later using the <code>hadbm refragment</code> command. For details about refragmentation, see <a href="#">“Refragmenting the HADB” on page 536</a>.</p> <p>You <i>must</i> use this option if you created the database using <code>--inetd</code>. In this case, you must refragment the database in a separate step using <code>hadbm refragment</code>.</p> <p>If you do not have sufficient device space for a refragmentation, you can recreate the database with more nodes. See <a href="#">“Adding Nodes Without Refragmenting” on page 537</a>.</p>
<code>--spares</code>	<code>-s</code>	0	<p>Specifies the number of new spare nodes in addition to those that already exist. This number must be even and must not be greater than the number of nodes added. Spare nodes are optional, but having two or more ensures high availability.</p>

---

**Table 21-11** `hadbm addnodes` Options (*Continued*)

Long Form	Short Form	Default	Description
<code>--dbpassword</code>	<code>-p</code>	none	Specifies the HADB system user password. You can use <code>--dbpasswordfile</code> instead. For details, see <a href="#">“The hadbm Command” on page 510</a> .
<code>--dbpasswordfile</code>	<code>-P</code>	none	Specifies a file that stores the password for the HADB system user. For details, see <a href="#">“The hadbm Command” on page 510</a> .
<code>--inetdsetupdir</code>	<code>-u</code>	current directory	Specifies the directory in which to store the <code>inetd</code> setup files. The directory must exist on the machine and must be writable.
<code>--hosts</code>	<code>-H</code>	none	<p>Specifies a comma-separated list of new host names for the new nodes in the database. One node is created for each comma-separated item in the list. The number of nodes must be even.</p> <p>Using duplicate host names creates multiple nodes on the same machine with different port numbers. Make sure that nodes on the same machine are not mirror nodes.</p> <p>Odd numbered nodes are in one DRU, even numbered nodes in the other. If <code>--spares</code> is used, new spare nodes are those with the highest numbers.</p> <p>If the database was created with double network interfaces, the new nodes must be configured in the same way. See <a href="#">“Configuring Double Networks” on page 520</a>.</p>
<code>dbname</code>	none	hadb	Specifies the database name. The database must be in the HA Fault Tolerant or Fault Tolerant state. For more information about database states, see <a href="#">“Getting the Status of the HADB” on page 539</a> .

## Refragmenting the HADB

You must refragment the database before new nodes can store data. Refragmentation is required to store data evenly across all active nodes. To refragment the database, use the `hadbm refragment` command. The syntax is as follows:

```
hadbm refragment --dbpassword=password | --dbpasswordfile=file [dbname]
```



For example:

```
hadbm refragment --dbpassword secret123
```

Refragmentation requires that the user data size not exceed 50% of the space available for user data. For details, see [“Getting Device Information” on page 542](#).

If this command fails even after multiple attempts, see [“Adding Nodes Without Refragmenting” on page 537](#).

The `hadbm refragment` command options are listed in the following table.

**Table 21-12** `hadbm refragment` Options

Long Form	Short Form	Default	Description
<code>--dbpassword</code>	<code>-p</code>	none	Specifies the HADB system user password. You can use <code>--dbpasswordfile</code> instead. For details, see <a href="#">“The hadbm Command” on page 510</a> .
<code>--dbpasswordfile</code>	<code>-P</code>	none	Specifies a file that stores the password for the HADB system user. For details, see <a href="#">“The hadbm Command” on page 510</a> .
<i>dbname</i>	none	hadb	Specifies the database name. The database must be in the HA Fault Tolerant or Fault Tolerant state. For more information about database states, see <a href="#">“Getting the Status of the HADB” on page 539</a> .

## Adding Nodes Without Refragmenting

If you don't refragment the database when adding nodes, you must clear the database and recreate the session store instead, otherwise the session store can't use the new nodes. You should not add nodes without refragmenting the database unless you can tolerate losing all data stored in the database. However, it may be the best alternative if all of the following conditions are met:

- You don't have enough disk space to expand each node as described in [“Adding Storage Space to Existing Nodes” on page 533](#).
- The user data size exceeds 50% of the space available for user data, which means you cannot refragment as described in [“Refragmenting the HADB” on page 536](#).
- You are not passivating the session state.

To add nodes without refragmenting, perform the following tasks:

1. Perform the following tasks for each server instance:
  - a. Disable the server instance in the load balancer, as described in [“Known Issues in Load Balancing Requests” on page 432](#).
  - b. Disable session persistence as described in [“Enabling and Disabling Availability” on page 474](#).
  - c. Restart the server instance.
  - d. Re-enable the server instance in the load balancer.

If you do not need to maintain availability, you can disable and reenable all the server instances at once in the load balancer. This saves time and prevents failover of outdated session data.

2. Stop the database as described in [“Stopping the HADB” on page 529](#).
3. Delete the database as described in [“Removing a Database” on page 532](#).
4. Recreate the database with the additional nodes as described in [“Creating a Database” on page 515](#).
5. Reconfigure the JDBC connection pool as described in [“Setting Up the JDBC Connection Pool” on page 522](#). You can also use the `cladmin` command. See Appendix F, [“Using the cladmin Command for Administration \(Enterprise Edition\).”](#)
6. Reload the session persistence store as described in [“Creating the Session Store” on page 468](#). You can also use the `cladmin` command.
7. Perform the following tasks for each server instance:
  - a. Disable the server instance in the load balancer.
  - b. Enable session persistence as described in [“Enabling and Disabling Availability” on page 474](#).
  - c. Restart the server instance.
  - d. Re-enable the server instance in the load balancer.

If you do not need to maintain availability, you can disable and reenable all the server instances at once in the load balancer. This saves time and prevents failover of outdated session data.

# Monitoring the HADB

You can monitor the activities in the HADB by performing the following tasks:

- [Getting the Status of the HADB](#)
- [Getting Device Information](#)

These sections briefly describe the `hadbm status`, `hadbm deviceinfo`, and `hadbm resourceinfo` commands. For details about interpreting HADB information, see the *Sun Java System Application Server Performance Tuning Guide*.

## Getting the Status of the HADB

To display the status of the database or its nodes, use the `hadbm status` command. The syntax is as follows:

```
hadbm status [--nodes] [dbname]
```

For example:

```
hadbm status --nodes
```

The physical node number is associated with a specific database node and port number combination, and does not vary during the life of the database. The logical node number, on the other hand, can vary during the lifetime of the database. Initially, logical node numbers are identical to physical node numbers for active nodes used to store data. Logical node numbering can change if individual nodes are stopped (for example, for maintenance), and spare nodes take over.

The `hadbm status --nodes` command gives information about both physical and logical node numbers. All other `hadbm` subcommands deal with physical node numbers only. You only need to know about logical node numbers if you need to know which nodes are currently mirror nodes. This information is useful when you are performing maintenance on machines. See [“Maintaining the HADB Machines” on page 544](#).

The `hadbm status` command options are listed in the following table.

**Table 21-13** `hadbm status` Options

Long Form	Short Form	Default	Description
<code>--nodes</code>	<code>-n</code>	not present	If present, displays node status information. See <a href="#">“Node Status” on page 540</a> .
<code>dbname</code>	<code>none</code>	<code>hadb</code>	Specifies the database name.

### Database Status

The possible states of a database are as follows:

- High-Availability Fault Tolerant (HAFT) - The database is fault tolerant and has at least one spare node on each DRU.
- Fault Tolerant (FT) - All the mirrored node pairs are up and running.
- Operational (O) - At least one node in each mirrored node pair is running.
- Non Operational (NO) - One or more mirrored node pairs is missing both nodes.
- Stopped (S) - No nodes are running in the database.
- Unknown (U) - The command cannot determine the state of the database.

If the database is Non Operational, clear the database using `hadbm clear` as described in [“Clearing the HADB” on page 531](#).

### Node Status

If you specify the `--nodes` option, the following information is displayed for each node in the database:

- Node number
- Name of the machine where the node is running
- Port number of the node
- Role of the node. For a list of possible roles and their meanings, see [“Roles of a Node” on page 541](#).
- State of the node. For a list of possible states and their meanings, see [“States of a Node” on page 541](#).
- Number of the corresponding mirror node.

A node's role and state can change as described in these sections:

- [Roles of a Node](#)
- [States of a Node](#)

### *Roles of a Node*

A node is assigned a role during its creation and can take any one of these roles:

- **Active:** An active node allows data storage and client access. Active nodes are in mirrored pairs.
- **Spare:** After having their data devices initialized, spare nodes monitor other data nodes to initiate repair if another node becomes unavailable. A spare node allows client access, but not data storage.
- **Offline:** A node is taken offline prior to stopping it to prevent restart by `inetd`. Offline nodes provide no services until their role changes. An offline node's role can change back to its former role.
- **Shutdown:** An intermediate step between active and offline, which a node occupies while waiting for a spare node to take over its functioning. After the spare node has taken over, the node is taken offline.

### *States of a Node*

A node can be in any one of the following states:

- **Starting:** The node is starting.
- **Waiting:** The node cannot decide its start level and is offline. If a single node is in this state for more than two minutes, stop the node and then start it at the `repair` level; see [“Stopping a Node” on page 527](#) and [“Starting a Node” on page 526](#). If multiple nodes are in this state, clear the database as described in [“Clearing the HADB” on page 531](#).
- **Running:** The node is providing all services that are appropriate for its role.
- **Stopping:** The node is in the process of stopping.
- **Stopped:** The node is inactive. Repair of a stopped node is prohibited.
- **Recovering:** The node is being recovered. When a node fails, the mirror node takes over the functions of the failed node. The failed node tries to recover by using the data and log records in main memory or on disk. The failed node uses the log records from the mirror node to catch up with the transactions performed when it was down. If recovery is successful, the node becomes active. If recovery fails, the node state changes to `Repairing`.

- **Repairing:** The node is being repaired. This operation reinitializes the node and copies the data and log records from the mirror node. Repair is more time consuming than recovery.

## Getting Device Information

Monitoring the HADB involves making sure that there is enough free space for the growth of the database. To get information about disk storage devices on each active node, use the `hadbm deviceinfo` command. The syntax is as follows:

```
hadbm deviceinfo [--details] [dbname]
```

For example:

```
hadbm deviceinfo --details
```

The default *dbname* is `hadb`.

The information displayed for each node of the database includes:

- Total device size (allocated in MB).
- Free size (in MB).
- Usage (in percent).

To determine the space available for user data, take the total device size, then subtract 4 times the `LogBufferSize`. If you do not know the size of the log buffer, use the command `hadbm get logbufferSize`. For example, if the total device size is 128 MB and the `LogBufferSize` is 24 MB, the space available for user data is  $128 - (4 \times 24) = 32$  MB.

The difference between the total device size and the free size is the user data size. If the data may be refragmented in the future, the user data size should not exceed 50% of the space available for user data. If refragmentation is not relevant, close to 100% may be used. Resource consumption warnings are written to the history files when the system is running short on device space.

For more information about tuning the HADB, see the *Sun Java System Application Server Performance Tuning Guide*.

If the `--details` option is specified, additional information is displayed:

- Number of reads.
- Number of writes.
- Name of the device.

For example:

NodeNO	Totalsize	Freesize	Usage	NReads	NWrites	DeviceName
0	128	120	6%	10000	5000	/var/opt/hadb.data.0
1	128	124	3%	10000	5000	/var/opt/hadb.data.1
2	128	126	2%	9500	4500	/var/opt/hadb.data.2
3	128	126	2%	9500	4500	/var/opt/hadb.data.3

If you need additional information, you can use the `hadbm resourceinfo` command. This command displays HADB runtime resource information that helps to identify resource contention, which you can use to reduce performance bottlenecks. For details, see the *Sun Java System Application Server System Deployment Guide* and the *Sun Java System Application Server Performance Tuning Guide*. The syntax is as follows:

```
hadbm resourceinfo [--databuf] [--locks] [--logbuf] [--nilogbuf] [dbname]
```

The following database information is displayed based on the options you specify:

- `--databuf` displays the data buffer pool information.
- `--locks` displays the locks information.
- `--logbuf` displays the log buffer information.
- `--nilogbuf` displays the node internal log device information.

For example, data buffer pool information is as follows:

NodeNO	Avail	Free	Access	Misses	Copy-on-Write
0	256	128	100000	50000	1000
1	256	128	110000	45000	950

Locks information is as follows:

- Node Number
- Available
- Free
- Waits

For example:

NodeNO	Avail	Free	Waits
0	50000	20000	10
1	50000	20000	0

No more than 50% of the allocated locks are used for primary recording operations. The other 50% are reserved for hot standby recording operations. To change the `NumberOfLocks`, see [“Viewing and Modifying Configuration Attributes” on page 546](#).

Log buffer information is as follows:

- Node Number
- Available: amount of memory allocated for the log buffer in MB
- Free: amount of free memory in MB

For example:

NodeNO	Avail	Free
0	16	2
1	16	3

Node internal log device information is as follows:

- Node Number
- Available: amount of memory allocated for the log device in MB
- Free: amount of free memory in MB

For example:

NodeNO	Avail	Free
0	16	2
1	16	3

## Maintaining the HADB Machines

The HADB achieves fault tolerance by replicating data on mirror nodes. Mirror nodes should be placed on separate DRUs in a production environment as described in [“HADB Architecture” on page 509](#).



A failure is an unexpected event such as a hardware failure, power failure, or operating system reboot. The HADB tolerates single failures: of one node, one machine (that has no mirror node pairs), one or more machines belonging to the same DRU, or even one entire DRU. However, the HADB does *not* automatically recover from a double failure, which is the simultaneous failure of one or more mirror node pairs. If a double failure occurs, you must clear the HADB and recreate its session store, which erases all its data.

Installing the entire HADB on a single machine is recommended *only* for development and test environments, because in this case any failure except a single node failure is a double failure.

---

**CAUTION** Before performing any maintenance, make sure you know which nodes are mirror nodes so you don't shut down a mirror node pair and make the database Non Operational. See [“Getting the Status of the HADB” on page 539](#).

---

Otherwise, to perform planned or unplanned maintenance on a single machine without interrupting HADB service:

1. For planned maintenance, stop all nodes on the machine. See [“Stopping a Node” on page 527](#).
2. Perform the maintenance procedure and get the machine up and running.
3. Start all nodes on the machine if either of the following is true:
  - If you stopped all nodes manually in [Step 1](#), regardless of whether you are using `inetd`
  - If you are not using `inetd`, regardless of how the nodes were stopped
 See [“Starting a Node” on page 526](#).
4. Check whether the nodes are active and running. See [“Getting the Status of the HADB” on page 539](#).

To perform planned maintenance on all HADB machines without interrupting HADB service:

1. For each spare machine in the first DRU, repeat the single machine procedure, one by one, for each machine.
2. For each active machine in the first DRU, repeat the single machine procedure, one by one, for each machine.
3. Repeat [Step 1](#) and [Step 2](#) for the second DRU.

To perform planned maintenance with HADB service interruption on all HADB machines, or when the entire HADB is on a single machine:

1. Stop the HADB. See [“Stopping the HADB” on page 529](#).
2. Perform the maintenance procedure and get all the machines up and running.
3. Start the HADB. See [“Starting the HADB” on page 529](#). The data stored in the database before the stop is available again.

To perform unplanned maintenance in the event of a failure, first check the database status. See [“Getting the Status of the HADB” on page 539](#).

- If the database state is Operational or better, this means the machines needing unplanned maintenance *do not* include mirror nodes. Follow the single machine procedure for each failed machine, one DRU at a time. HADB service is not interrupted.
- If the database state is Non-Operational, this means the machines needing unplanned maintenance include mirror nodes. One such case is when the entire HADB is on a single failed machine. Get all the machines up and running first. Then clear the HADB and recreate the session store. See [“Clearing the HADB” on page 531](#) and [“Creating the Session Store” on page 468](#). This interrupts HADB service.

## Viewing and Modifying Configuration Attributes

You can modify database configuration attributes. This section describes the following tasks:

- [Getting the Values of Configuration Attributes](#)
- [Setting the Values of Configuration Attributes](#)
- [Configuration Attributes](#)

### Getting the Values of Configuration Attributes

To get the values of configuration attributes (for a list, see [“Configuration Attributes” on page 548](#)), use the `hadbm get` command. The syntax is as follows:

```
hadbm get attribute-list | --all [dbname]
```

For example:

```
hadbm get JdbcUrl,NumberOfSessions
```

The default *dbname* is `hadb`. The *attribute-list* is a comma-separated or quote-enclosed space-separated list of attributes. The `--all` option displays values for all attributes.

## Setting the Values of Configuration Attributes

To set the values of configuration attributes (for a list, see [“Configuration Attributes” on page 548](#)), use the `hadbm set` command. The syntax is as follows:

```
hadbm set [dbname] attribute=value,attribute=value ...
```

The default *dbname* is `hadb`. The *attribute-list* is a comma-separated or quote-enclosed space-separated list of attributes.

If execution of this command is successful, the database is restarted in the state it was in previously, or in a better state. For information about database states, see [“Getting the Status of the HADB” on page 539](#).

If execution of this command is unsuccessful, restart the HADB as described in [“Restarting the HADB” on page 530](#).

The following attributes cannot be set by `hadbm set`, but can be set during database creation using `--set` or other options of `hadbm create`: `ConfigPath`, `DatabaseName`, `DevicePath`, `HistoryPath`, `InstallPath`, `ManagementProtocol`, `NumberOfDataDevices`, and `Portbase`. For information about `hadbm create`, see [“Creating a Database” on page 515](#).

The `JdbcUrl` attribute value is derived from the `--hosts` and `--portbase` options during database creation with `hadbm create` and cannot be set by `hadbm set` or the `--set` option.

All other attributes listed in [Table 21-14](#) can be set using `hadbm set`.

---

<b>NOTE</b>	Using <code>hadbm set</code> to set any configuration attribute except <code>ConnectionTrace</code> or <code>SQLTraceMode</code> causes a rolling restart of the HADB. In a rolling restart, each node is stopped and started one at a time. If you set <code>ConnectionTrace</code> or <code>SQLTraceMode</code> , no rolling restart occurs, but the change only takes effect for new HADB connections made from a Sun Java System Application Server instance.
-------------	---

---

# Configuration Attributes

The following table lists the configuration attributes that you can get and set. Except where noted, sizes are in MB, and times are in seconds.

**Table 21-14** Configuration Attributes

Attribute	Default	Range	Description
ConfigPath	/etc/opt/SUNWhadb/ dbdef		(get only) Location of configuration files used internally by the HADB.
ConnectionTrace	FALSE		If true, records a message in the HADB history files when a client connection (JDBC, ODBC) is initiated or terminated.
CoreFile	FALSE		The default value should not be changed.
DatabaseName	hadb		(get only) Name of the database.
DataBufferPoolSize	200	16MB - 2GB	Size of the data buffer pool allocated in shared memory.
DataDeviceSize	none	Maximum is the smaller of 256 GB or the maximum operating system file size, minimum is:  (4 x LogbufferSize + 16MB) / NumberOfData devices  HADB uses this amount of space internally. Therefore, this space is not available for storing user data.	The size of each data device used by an HADB node.  DataDeviceSize is equal to:  TotalDatadeviceSizePerNode / NumberOfDatadevices  Therefore,  TotalDatadeviceSizePerNode and DataDeviceSize are mutually dependent: changing one changes the other.  The recommended size is four times the expected size of the user data, based on the number of users and the size of each user record.

**Table 21-14** Configuration Attributes (*Continued*)

Attribute	Default	Range	Description
DevicePath	/var/tmp		(get only) Location of the devices. There are three devices: the DataDevice, the NiLogDevice (node internal log device), and the RelalgDevice (relational algebra query device).
EagerSessionThreshold	50 (% of NumberOfSessions)	0 - 100	<p>Determines whether normal or eager idle session expiration is used.</p> <p>In normal idle session expiration, sessions that are idle for more than SessionTimeout seconds are expired.</p> <p>When the number of concurrent sessions exceeds the EagerSessionThreshold percentage of the maximum number of sessions, sessions that are idle for more than EagerSessionTimeout seconds are expired.</p>
EagerSessionTimeout	120	0 - 2000000	Amount of time a database connection can be idle before it expires when eager session expiration is used.
EventBufferSize	0	0MB - 2GB	<p>Size of the event buffer, where database events are logged. If set to 0, no event buffer logging is performed.</p> <p>During failures, the event buffer is dumped. This gives valuable information on the cause of the failures and is useful during trial deployment.</p> <p>Writing events to memory has a performance penalty.</p>
HistoryPath	/var/tmp		(get only) Location of the HADB history files, which contain information, warnings, and error messages.

**Table 21-14** Configuration Attributes (*Continued*)

Attribute	Default	Range	Description
InstallPath	parent of the directory where <code>hadbm</code> resides: <i>install_dir</i> /SUNWhadb/4/		(get only) Location of the HADB executable files.
InternalLogbufferSize	12	4MB - 128MB	Size of the node internal log device, which keeps track of operations related to storing data.
JdbcUrl	none		(get only) The JDBC connection URL for the database.
LogbufferSize	48	4MB - 2GB	Size of the log buffer, which keeps track of operations related to data.
ManagementProtocol	ssh	rsh, ssh	(get only) Protocol used between the <code>hadbm</code> command and the database nodes.
MaxTables	1100	100 - 1100	Maximum number of tables allowed in an HADB database.
NumberOfDatadevices	1	1 - 8	(get only) Number of data devices used by an HADB node.
NumberOfLocks	50000	20000 - 2147483647	Number of locks allocated by an HADB node.
NumberOfSessions	100	1 - 10000	Maximum number of sessions (database connections) that can be opened for an HADB node.
Portbase	15200	10000 - 63000	(get only) Base port number used to create different port numbers for different HADB processes.
RelalgdeviceSize	128	32MB - 2GB	Size of the device used in relational algebra queries.
SessionTimeout	1800	0 - 2000000	Amount of time a database connection can be idle before it expires when normal session expiration is used.
SQLTraceMode	NONE	NONE, SHORT, FULL	Amount of information about executed SQL queries written to the history files.  If <code>SHORT</code> , login and logout of SQL sessions are recorded. If <code>FULL</code> , all SQL queries being prepared and executed, including parameter values, are recorded.

**Table 21-14** Configuration Attributes (*Continued*)

Attribute	Default	Range	Description
StartRepairDelay	20	0 - 100000	Maximum time a spare node allows for a failed active node to perform a node recovery. If the failed node cannot recover within this time interval, the spare node starts copying data from the failed node's mirror and becomes active. Changing the default value is not recommended.
StatInterval	600	0 - 1000	<p>Interval at which an HADB node writes throughput and response time statistics to its history file. To disable, set to 0.</p> <p>An example statistics line is as follows:</p> <pre>"Req-reply time: # 123, min= 69 avg= 1160 max= 9311 %=100.0"</pre> <p>The # is the number of requests serviced over the StatInterval. The next three numbers are the minimum, average, and maximum time in microseconds taken by transactions completed over the StatInterval. The % is the number of transactions completed successfully within 15 milliseconds over the StatInterval.</p>
SyslogFacility	local0	local0, local1, local2, local3, local4, local5, local6, local7, kern, user, mail, daemon, auth, syslog, lpr, news, uucp, cron, none	<p>Facility used when reporting to syslog (see <code>man syslog</code> for details). The <code>syslog</code> daemon should be configured (see <code>man syslogd.conf</code> for details).</p> <p>Use a facility that is not in use by other applications running on the same machine.</p> <p>Set to <code>none</code> to disable <code>syslog</code> logging.</p>

**Table 21-14** Configuration Attributes (*Continued*)

Attribute	Default	Range	Description
SysLogging	TRUE		Set to true if an HADB node should write any information to the operating system's <code>syslog</code> files.
SysLogLevel	warning	none, alert, error, warning, info	Lets you to fine tune which HADB messages are reported to the operating system's <code>syslog</code> files.  Higher levels include lower levels. For example, the <code>error</code> level includes <code>alert</code> messages, and the <code>info</code> level includes all messages.
SyslogPrefix	HADB		Text string that is prepended to all <code>syslog</code> messages written by the HADB.
TakeoverTime	10000 (milliseconds)	500 - 16000	Time between when a node fails and when its mirror takes over. Changing the default value is not recommended.
TotalDatadeviceSizePerNode	none	Maximum is the smaller of 256 GB or the maximum operating system file size, minimum is:  4 x LogbufferSize + 16MB  HADB uses this amount of space internally. Therefore, this space is not available for storing user data.	Sum of the sizes of all data devices used by an HADB node.  <code>TotalDatadeviceSizePerNode</code> is equal to:  <code>DataDeviceSize</code> x <code>NumberOfDatadevices</code>  Therefore, <code>TotalDatadeviceSizePerNode</code> and <code>DataDeviceSize</code> are mutually dependent: changing one changes the other.



# Clearing and Archiving History Files

HADB history files contain a record of database operations and error messages. The location of these files is determined by the `--historypath` option of the `hadbm create` command. The default location is `/var/tmp`. These files have names of the format `dbname.out.nodeno`. For details about `hadbm create`, see [“Creating a Database” on page 515](#).

These history files grow over time. To save space and prevent files from getting too large, you should periodically clear and archive older history files. To clear the history files of a database, use the `hadbm clearhistory` command. The syntax is as follows:

```
hadbm clearhistory [--saveto=path] [dbname]
```

The default *dbname* is `hadb`.

Use the `--saveto` or `-o` option to specify a directory if you want to store the old history files. This directory must have write permissions set.

Each message in the history file contains the following information:

- The name, in abbreviated form, of the HADB process that produced the message.
- The type of message. Possible values are as follows:
  - INF - general information
  - WRN - warnings
  - ERR - errors
  - DBG - debug information
- A timestamp. The time is obtained from the system clock on the machine used for the node.

Messages about resource shortages contain `HIGH LOAD`.

You do not need a detailed knowledge of all the various types of entries in the history file. If for any reason you need to study a history file in greater detail, you should obtain help from Sun customer support. See [“Using Sun Customer Support for the HADB” on page 554](#).

# Recovering from Session Data Corruption

The following are indications that session data may be corrupted:

- Error messages appear in the Sun Java System Application Server system log (server log) every time you try to save the session state.
- Error messages are written to the server log indicating that the session could not be found or could not be loaded during session activation.
- Sessions that are activated after previously being passivated contain empty or incorrect session data.
- When an instance fails, failed-over sessions contain empty or incorrect session data.
- When an instance fails, instances that try to load a failed-over session cause an error in the server log indicating the session could not be found or could not be loaded.

To bring the session store back to a consistent state if you determine that the data has been corrupted, do the following:

1. Clear the session store. For more information, see [“Clearing the Session Store” on page 470](#).
2. If clearing the session store doesn’t work or you continue to see errors in the server log, reinitialize the data space on all the nodes and clear the data in the database. See [“Clearing the HADB” on page 531](#).
3. If clearing the database doesn’t work, delete and then recreate the database. See [“Removing a Database” on page 532](#) and [“Creating a Database” on page 515](#).

## Using Sun Customer Support for the HADB

Before calling Sun customer support about HADB issues, you should gather as much of the following information about your system as possible:

- Usage profile
  - Number of active concurrent users
  - Number of passive users
  - Number of users entering the system per second

- Average session length and size
- Session state timeout period (`SessionTimeout` value)
- Transaction rate per user per second
- Machine properties
  - Physical RAM size
  - Number of CPUs
  - CPU speed
  - Operating system version
  - Number of physical disks
  - Total disk size
  - Available disk space
  - Data transfer capacity
- Network properties
  - Transfer capacity
  - Number of host names (network interfaces) per node
- HADB data
  - History files
  - `cfg` and `def` files, located in the *configpath*/`dbdef`/`hadb` directory
  - Version information (`hadbm --version`)

# Environment Variables

This table lists environment variables that correspond to `hadbm` command options.

**Table 21-15** HADB Options and Environment Variables

Long Form	Short Form	Default	Environment Variable
<code>--configpath</code>	<code>-c</code>	<code>/etc/opt/SUNWhadb</code>	<code>\$HADB_CONFIGPATH</code>
<code>--datadevices</code>	<code>-a</code>	<code>1</code>	<code>\$HADB_DATADEVICES</code>
<i>dbname</i>	<code>none</code>	<code>hadb</code>	<code>\$HADB_DB</code>

**Table 21-15** HADB Options and Environment Variables (*Continued*)

Long Form	Short Form	Default	Environment Variable
--dbpassword	-p	none	\$HADB_DBPASSWORD
--dbpasswordfile	-F	none	\$HADB_DBPASSWORDFILE
--devicepath	-d	/var/opt/SUNWhadb	\$HADB_DEVICEPATH
--devicesize	-z	none	\$HADB_DEVICESIZE
--echo	-e	FALSE	\$HADB_ECHO
--fast	-F	FALSE	\$HADB_FAST
--force	-f	FALSE	\$HADB_FORCE
--help	-?	FALSE	\$HADB_HELP
--historypath	-t	/var/tmp	\$HADB_HISTORYPATH
--hosts	-H	none	\$HADB_HOSTS
--inetd	-I	FALSE	\$HADB_INETD
--inetdsetupdir	-u	current directory	\$HADB_INETDSETUPDIR
--installpath	-n	parent of the directory where hadbm resides: <i>install_dir</i> /SUNWhadb/4/	\$HADB_INSTALLPATH
--interactive	-i	TRUE	\$HADB_INTERACTIVE
--no-refragment	-r	FALSE	\$HADB_NOREFRAGMENT
--portbase	-b	15200	\$HADB_PORTBASE
--quiet	-q	FALSE	\$HADB_QUIET
--repair	-R	TRUE	\$HADB_REPAIR
--rolling	-g	TRUE	\$HADB_ROLLING
--saveto	-o	none	\$HADB_SAVETO
--set	-S	none	\$HADB_SET
--spares	-s	0	\$HADB_SPARES
--startlevel	-l	normal	\$HADB_STARTLEVEL
--version	-V	FALSE	\$HADB_VERSION
--yes	-y	FALSE	\$HADB_YES

# Appendixes

Appendix A, “Using the Command Line Interface”

Appendix B, “Failover Scenarios (Enterprise Edition)”

Appendix C, “Compiling and Configuring Apache Web Server  
(Enterprise Edition)”

Appendix D, “Running Message Queue Broker with Trial  
Enterprise License”

Appendix E, “HTTP Load Balancer Configuration File”

Appendix F, “Using the cladadmin Command for Administration  
(Enterprise Edition)”



# Using the Command Line Interface

This appendix provides instructions for using the command line interface (the `asadmin` utility), in `singlemode` (that is, you run one command at a time from the command prompt) at the system prompt, in `multimode` (that is, you can run multiple commands without needing to reenter environment-level information), and in scripts and programs. You can use the command line interface in place of the Administration interface screens.

This appendix contains the following sections:

- [About the Command Line Interface](#)
- [Using `asadmin`](#)
- [Security Considerations](#)
- [Concurrent Access Considerations](#)
- [Command Reference](#)

## About the Command Line Interface

This section contains the following topics:

- [About the `asadmin` Utility](#)
- [About Ant Tasks](#)
- [About Other Command Line Utilities](#)

## About the asadmin Utility

The `asadmin` utility performs all configuration and administration tasks. You can use this utility in place of using the Administration interface.

## About Ant Tasks

Many developers use Ant to accelerate the development process for J2EE applications. Ant scripts leverage the `asadmin` utility for some tasks. Developers use the Ant tasks for building applications, deploying and undeploying modules and applications and for controlling the Sun Java System Application Server.

For more information on the Ant tasks, see the *Sun Java System Application Server Developer's Guide*.

For more information on Ant, see the Jakarta Project site at <http://jakarta.apache.org/ant/>.

## About Other Command Line Utilities

Sun Java System Application Server contains additional command line utilities. The following table lists the utilities and gives a brief description of each one.

**Table A-1** Other Command-Line Utilities

Utility	Definition
appclient	Launches the Application Client Container and invokes the client application packaged in the application JAR file.
capture-schema	Gets the database schema and mapping information.
flexanlg	Generates statistics about your server.
htpasswd	Creates the user authentication files.
package-appclient	Packs the application client container libraries and jar files. For more information, see the <i>Sun Java System Application Server Developer's Guide to Clients</i> .
verifier	Validates the J2EE deployment descriptors with the DTDs. For more information, see the <i>Sun Java System Application Server Developer's Guide</i> .
wscompile	Takes the service definition interface and generates the client stubs or server-side skeletons; or generates a set of WSDL for the provided interface.
wsdeploy	Generates a deployable WAR file.



**Table A-1** Other Command-Line Utilities (*Continued*)

Utility	Definition
clsetup	Automates the process of setting up a cluster in a typical single machine configuration. (Enterprise Edition only)
cladmin	Runs certain <code>asadmin</code> commands simultaneously on all application server instances in a cluster. (Enterprise Edition only)

For more information on these utilities, see their online help.

## Using asadmin

The `asadmin` utility has a set of commands for performing administrative tasks. You can use these commands to most of the tasks that you can perform using the Administration interface. You can find the `asadmin` utility at `install_dir/bin` and run it from there. On Windows, double-click the `asadmin.bat` file and the `asadmin` utility launches in a command window, running in multimode.

Please note that some HTTP-server-related properties and the Administration Server properties cannot be set using the command line; they must be set using the Administration interface. You can set all properties stored in the `server.xml` configuration file, but not the ones stored in `init.conf` or `obj.conf`. For more information on the configuration files, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

For more information on individual commands, see [“Command Reference” on page 579](#) and the help for the commands.

This section contains the following topics:

- [Understanding the Command Syntax](#)
- [Using Singlemode and Multimode](#)
- [Using Interactive and Non-Interactive Options](#)
- [Using the Environment Commands](#)
- [Using the Password File Option](#)
- [Running asadmin Locally or Remotely](#)
- [Using Command Line Invocations](#)
- [Using Escape Characters](#)

- [Using get and set Commands](#)
- [Using Help](#)
- [Viewing Output and Errors](#)

## Understanding the Command Syntax

The `asadmin` utility has the following syntax:

*asadmin command -short-option argument --long-option argument operand*

### Command

The command is the operation or task performed. The command is case-sensitive.

### Options

Options modify how the utility performs a command. Options are case-sensitive. Notice that the short options have a single dash (-) in front of them, and the long options have two dashes (--) in front of them. For many options, you can use either a long or short form; for example, user can be either `--user` or `-u`. Some options are required and some are optional. Optional options are shown in brackets in a command's syntax. You must include all required options when you run a command or you receive an error message and the command does not execute.

For a list of available long and short option names available see [Table A-40](#) in “[Command Reference](#)” on page 579.

Most options require argument values, for example `--port port_number`. The exceptions are boolean options, which toggle to switch a feature on or off and don't require argument values.

You can also save options in the environment variables. For more information, see “[Using the Environment Commands](#)” on page 565. For a complete list of the environment variable equivalent of options, see “[Long and Short Option Formats, Default Values, and Environment Variable Equivalents](#)” on page 611.

### Boolean Options

A boolean options toggles on or off (for example `--interactive` puts you in interactive mode where you are prompted for options; `--no-interactive` turns off interactive mode). Putting `--no-` in front of the long option toggles the option off. Specifying the short option name always sets the opposite of the default value.

You can group short boolean options. For example, you can use `-Ie` to specify interactive (short option `-I`) and echo (short option `-e`).

## Operands

Operands are set off by a space or a tab. They can come in any order in the command syntax. You can use `--` with no option after it to separate the options from the operands. Any following arguments are treated as operands, even if they begin with a dash (`-`). For example, in

```
asadmin> create-jvm-options --instance server1 -- -Xmx1500m
```

`-Xmx1500m` is treated as an operand, even though it begins with a dash.

## Syntax Example

```
asadmin create-instance [--user admin_user] [--password admin_password] [-H
host_name] [--port port_number] [--sysuser sys_user] [--domain domain_name]
[--local=true/false] [--passwordfile file_name] [--secure | -s] --instanceport
instance_port instance_name
```

In this syntax example `-H` is the short option for hostname, `--user` is a long option with *admin\_user* as its argument, and *instance\_name* is an operand. Optional options are inside brackets.

The following example shows the syntax with real values. Some of the optional options are not used in the example.

```
asadmin create-instance --user admin --password password -H austen --port
4848 --instanceport 1024 server2
```

## Using Singlemode and Multimode

You can run `asadmin` either in singlemode or multimode. In singlemode you run one command at a time from the command prompt. In multimode you can run multiple commands without needing to reenter environment-level information.

If you are using input from a file, and a command fails, in singlemode the program exits. If you are using multimode and a command fails, you return to the `asadmin` prompt.

## Singlemode

If you invoke a single command in the command line interface from a command prompt, you are running in singlemode. The command line interface runs the command and exits back to the command prompt. To run the command line interface from a command prompt, go to the directory *install\_dir/appserv/bin* and type your command at the command prompt:

```
> asadmin command options arguments
```

For example:

```
> asadmin create-instance --user admin --password password -H austen --port
4848 --instanceport 1024 server2
```

## Multimode

Multimode lets you set your environment at the outset so that you can run multiple commands without needing to re-enter certain environment-level information, such as your server name, port, and your password. One significant advantage of using multimode is that commands can be entered and executed significantly faster, since *asadmin* stays in memory. If these environment variables are set at the operating system level, multimode picks up those settings. The *asadmin* utility uses these settings until you change them.

On Windows, you are automatically in multimode when you run the *asadmin.bat* file.

On UNIX, to start the *asadmin* utility in multimode from the command line, type:

```
> asadmin multimode
```

When you are in multimode, the command prompt changes to *asadmin*. You can then type your commands into the *asadmin* prompt. You do not need to use the utility name. For example:

```
asadmin> create-instance --user admin --password password -H austen --port
4848 --instanceport 1024 server2
```

Exit multimode by typing *exit* or *quit*. You return to the command prompt.

## Multiple Multimode

You can also invoke multimode from within a multimode session, by using the command:

```
asadmin> multimode
```

Once you exit the second multimode environment, you return to your original multimode environment.

For example, if you are administering `server1` in multimode, and you want to administer `server2` to compare the two, you can invoke multimode for `server2` from within multimode for `server1`. Because you do not need to exit your current multimode session, you can retain your environment settings. When you exit the multimode session you're using for `server2`, you return to your `server1` multimode environment.

## Using Interactive and Non-Interactive Options

When you use the command line interface, you can use it in interactive or non-interactive mode. If you choose to use interactive mode, you are prompted for the password if you don't specify it. Interactive mode is enabled by default.

You can disable and enable interactive mode by setting the interactive environment variable using the `export` command. For more information, see [Table A-2 on page 567](#).

You can use the interactive option in singlemode under all circumstances. You can use the interactive option in multimode when you run one command at a time from the command prompt, and when you run in multimode from a file. However, commands in multimode when piped from an input stream and commands invoked from another program cannot run in interactive mode.

## Using the Environment Commands

The `asadmin` utility contains a set of environment variables that you can set using environment commands. In multimode, once you set these variables you do not need to reset your environment until you exit multimode. You can also set these environment variables at the operating system level; if you do, they are automatically picked up when you enter multimode and they persist after you exit from multimode.

Environment variables are name/value pairs that you can set at any time by assignment. Environment variable constitutes of `AS_ADMIN_` prefix to the option name in capitals. For example, to set the Administration Server user, you can type:

```
export AS_ADMIN_USER=administrator
```

Where *administrator* is the administrator's username.

This also makes the value for AS\_ADMIN\_USER available to asadmin commands; for example:

```
asadmin multimode
asadmin> export AS_ADMIN_HOST=austen
```

For as long as you are in this multimode session, the Administration Server hostname is set to austen, unless you reassign it.

You can also set and export values for multiple environment variables in one step, for example:

```
asadmin> export AS_ADMIN_PORT=4848 AS_ADMIN_USER=admin
```

To see your current environment variable settings, use the export command without arguments:

```
asadmin> export
AS_ADMIN_HOST=austen
AS_ADMIN_PORT=4848
AS_ADMIN_USER=admin
```

Use the unset command to remove a variable and its value from the environment. For example:

```
asadmin> unset AS_ADMIN_HOST
```

You can override the set value for an environment variable either by resetting the variable, or by setting a different value as part of an asadmin command. For example:

```
asadmin> export AS_ADMIN_HOST=dickens
asadmin> show-instance-status --host austen instance-name
```

This example shows the instance status for an instance in the Administration Server host austen, because that value overrides the earlier host value of dickens.

If you do not use exported variables, you must provide the following options with most commands or use the default values (for a list of default values, see [“Long and Short Option Formats, Default Values, and Environment Variable Equivalents”](#) on page 611):

- --host
- --port
- --user
- --password or --passwordfile

- `--secure=true` (if secure)
- `--instance` (if needed)

**Table A-2** describes some of the environment variables to use with the `export` command. These variables are the most commonly used, as they specifically deal with setting up the environment. The first column shows the environment variable name, and the second column shows what it is used for, and the default value if none is set. For a complete list of the environment variables, see “[Long and Short Option Formats, Default Values, and Environment Variable Equivalents](#)” on page 611.

**Table A-2** Environment Variables Used with `export` Command

Environment Variable	Use
AS_ADMIN_HOST	Hostname of the Administration Server. If no value is specified, uses <code>localhost</code> .
AS_ADMIN_PORT	Port number of the Administration Server. If no value is specified, uses 4848.
AS_ADMIN_USER	Username of the user to run the commands.
AS_ADMIN_PASSWORD	Password of the username that runs the commands. The username and password are used to authenticate the user, to verify that the user is allowed to administer the server. It is the same as the authentication that takes place when you access the Administration Server through the Administration interface.
AS_ADMIN_SECURE	<code>=true</code> if secure.
AS_ADMIN_INSTANCE	Sets the instance of the Sun Java System Application Server. Any subsequent commands that use the instance name as an argument (but not those that use it as an operand) use this specified instance.

## Using the Password File Option

If you do not want to type the passwords on the command line or set environment variables for the passwords, you can create a password file and use it as an option from the command line.

Every command which has the `a password` option also has the `passwordfile` option that you can use instead. The password file contains the following lines:

```
AS_ADMIN_PASSWORD=value
```

```
AS_ADMIN_ADMINPASSWORD=value
```

AS\_ADMIN\_USERPASSWORD=*value*

If you use the `passwordfile` option, the passwords in the file are exported to the multimode environment, and subsequent commands without the `password` options specified use these values.

If you specify both a password and a password file option at the command line, the values in the password file are exported to the multimode environment, but the current command uses the password specified in the password option, because the password option takes precedence over the password file.

## Running asadmin Locally or Remotely

Usually the `asadmin` utility sends its commands through the Administration Server. Therefore, you do not need to run `asadmin` on the system where the Sun Java System Application Server is installed. However, the Administration Server must be running in order for most `asadmin` commands to work.

Some commands have the option of being run locally, for example, `create-instance`. If you use the `--local=true` option with `create-instance`, you must run it on the machine where the server is installed, but you do not need to have the Administration Server running in order to create the instance.

Some commands must be run locally. For example, `start-appserv`, which starts the Administration Server and all its instances, cannot be run remotely because the Administration Server is not running until the command starts it.

For more information on the Administration Server, see [Chapter 2, “Setting Administration Server Preferences.”](#)

The following commands can be run both locally and remotely:

- `create-instance`
- `delete-instance`
- `list-instances`
- `start-instance`
- `stop-instance`
- `display-license`
- `version`
- `stop-domain`



- `restart-instance`
- `list-domains`

For these commands, you can choose to run a command locally without specifying the local option. By default, if you specify a value for user, password, host, or port in the command syntax the command is treated as a remote command (though you can still specify local values for these options). If you do not specify values for these options, the command is run locally by default.

When a command is executed locally, if there is a domain option it is required by the command (unless there is only one domain). When a command is executed remotely it ignores the domain option if you specify it.

## Using Command Line Invocations

You can invoke a command line in many ways, as described in the following topics:

- [Using asadmin from the Command Line](#)
- [Using asadmin with Input from a File \(Script\)](#)
- [Using asadmin with Standard Input \(Pipe\)](#)

### Using asadmin from the Command Line

The simplest way to use the commands is one at a time, from the command line. You type in the utility, command, its options and arguments. In multimode you type multiple commands without needing to retype the utility name and environment options (if you have set the environment variables). You can run either singlemode or multimode commands interactively (prompted for additional required input, for example, a password) or non-interactively.

For more information on singlemode and multimode, see [“Using Singlemode and Multimode” on page 563](#).

For more information on using commands interactively, see [“Using Interactive and Non-Interactive Options” on page 565](#).

#### *Example from Command Line*

```
> asadmin create-instance --user admin --password password --host austen
--port 4848 --instanceport 1024 server2
```

After the command is executed, the operating system prompt appears.

## Using asadmin with Input from a File (Script)

You can create a script which includes many `asadmin` commands. With scripts you can process commands in a batch, set up a job to run at specific times, and otherwise simplify and automate your administration tasks.

To call a script that is in a file, use the syntax:

```
> asadmin multimode --file filename
```

The following is an example of a simple script in a file that you can call in this manner:

```
# Create new instance and start it.
export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=mypassword
AS_ADMIN_HOST=austen AS_ADMIN_PORT=4848
create-instance --instanceport 9000 austen3
start-instance austen3
```

This script sets the environment, creates an instance called `austen3` and starts the new instance. Lines that begin with a number sign (#) are treated as comments and ignored.

## Using asadmin with Standard Input (Pipe)

You can pipe input into the `asadmin` utility using the following syntax:

```
cat filename | asadmin multimode
```

This syntax may not work on Windows.

## Using Escape Characters

Some characters, the colon (:), the asterisk (\*), and the backslash (\), cause errors if you use them in the command syntax unless you use escape characters to set them off. The possibilities for using escape characters vary depending upon what platform you use and whether you use singlemode or multimode.

---

<b>NOTE</b>	You do not need to use escape characters for colons in the <code>get</code> and <code>set</code> commands.
-------------	--

---

This section contains the following topics:

- [Escape Characters on UNIX in Singlemode](#)
- [Escape Characters on All Platforms in Singlemode](#)

- [Escape Characters on All Platforms in Singlemode](#)
- [Escape Characters on All Platforms in Multimode](#)

## Escape Characters on UNIX in Singlemode

On UNIX, you can use either two backslashes (\\) or double-quotes (“ ”) to escape restricted characters.

Note that this syntax does not work on Linux operating systems. For Linux, see [“Escape Characters on All Platforms in Singlemode” on page 572](#).

### *Escape with Backslashes (\)*

For example, when creating a JDBC connection pool with a option whose value includes colons, you could use backslashes (example assumes the environment variables have been set for some properties):

```
asadmin create-jdbc-connection-pool --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url=jdbc\:oracle\:thin\:@asperfsol8\:1521\:V8i:user=staging_lookup_ap
p:password=staging_lookup_app OraclePoollookup
```

### *Escape with Quotes*

To use quotes in the same example as above, you would enclose the value in double quotes (") and escape the double quotes with the backslash.

```
asadmin create-jdbc-connection-pool --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url=\"jdbc\:oracle\:thin\:@asperfsol8\:1521\:V8i\":user=staging_lookup_app:pass
word=staging_lookup_app OraclePoollookup
```

You can also use the methods described in [“Escape Characters on All Platforms in Singlemode” on page 572](#).

## Escape Characters on Windows in Singlemode

On windows, you can escape using the backslash character. For example, when creating a JDBC connection pool with a option whose value includes colons, you could use backslashes (example assumes the environment variables have been set for some properties):

```
asadmin create-jdbc-connection-pool --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url=jdbc\:oracle\:thin\:@asperfsol8\:1521\:V8i:user=staging_lookup_app:pas
sword=staging_lookup_app OraclePoollookup
```

You can also use the methods described in [“Escape Characters on All Platforms in Singlemode” on page 572](#).

## Escape Characters on All Platforms in Singlemode

On any platform, you can use backslashes to escape the character and enclose the value containing the escaped characters in double quotes. For example, when creating a JDBC connection pool with a option whose value includes colons, you could use the escape characters as follows (example assumes the environment variables have been set for some properties):

```
asadmin create-jdbc-connection-pool --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url="jdbc\:oracle\:thin\:@iasperfsol8\:1521\:V8i":user=staging_lookup_app:
password=staging_lookup_app OraclePoollookup
```

## Escape Characters on All Platforms in Multimode

In multimode you can use the following syntax, which only requires quotes, not slashes or backslashes:

```
asadmin> create-jdbc-connection-pool --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url="jdbc\:oracle\:thin\:@asperfsol8\:1521\:V8i":user=staging_lookup_app:passwo
rd=staging_lookup_app OraclePoollookup
```

## Using get and set Commands

Use the `get` and `set` commands to access and change configuration settings in the Sun Java System Application Server. In most cases, `asadmin` commands only set the *required* properties. Use the `set` command to change values for optional properties.

**Table A-3** get and set Commands

Command	Arguments	Use
get	(scope) where scope represents an attribute and is a valid name.	Gets the value of the attribute.
set	(scope= <i>value</i> ) where scope represents an attribute and is a valid name, and <i>value</i> is the value you want to set for that attribute.	Sets the value of the attribute.
reconfig	instance-name	After running any command that modifies the configuration files, you need to run reconfig in order for the changes to be applied to the server. For more information on applying changes/reconfiguring the server, see <a href="#">“Applying Changes to an Application Server Instance” on page 83</a> .

You can get or set multiple attribute values in a single command by using spaces between the attributes. For example:

```
set server1.appReloadPollInterval=20 server1.mime.mime1.file=mime.types
```

You can also use the `AS_ADMIN_PREFIX` environment variable to set a prefix that will be used by subsequent `get` and `set` commands. A period (“.”) is implicitly inserted between the prefix string and the operand in the `get` and `set` commands. For example:

```
asadmin>export AS_ADMIN_PREFIX=server1
asadmin>get *
server1.locale = en_US
server1.appReloadPollInterval = 2
server1.name = server1
...
```

Because the `get` and `set` commands require period separators, if an item contains periods in its name you must use the escape character backslash (\) before them. The following example shows a server instance name `server2.sun.com` with the periods preceded by backslashes:

```
get server2\.sun\.com.*
```

If you do not include the backslashes, you get an error message.

## get and set Command Examples

The following examples show how to use the `get` command to get the values of attributes, and how to use the `set` command to set values.

### *MDB Container Service Example*

If the application server instance is `server1`, you can get the values of all `mdb-container` attributes by using the following command in multimode, with your environment set:

```
asadmin> get server1.mdb-container.*
```

The following is an example of the output from this command, showing current values for the attributes:

```
server1.mdb-container.logLevel = null
server1.mdb-container.steadyPoolSize = 10
server1.mdb-container.idleInPoolTimeoutInSeconds = 600
server1.mdb-container.maxPoolSize = 60
server1.mdb-container.monitoringEnabled = false
server1.mdb-container.poolResizeQuantity = 2
```

To get just the value of the MDB container attribute `monitoringEnabled`, use the following:

```
asadmin> get server1.mdb-container.montioringEnabled
```

To set the value of the `monitoringEnabled` attribute to `true`, use the following:

```
asadmin> set server1.mdb-container.montioringEnabled=true
```

### *JMS Resource Example*

For configuring any of the resources, the attribute should look like:

*instancename.resource.primary\_key\_value.attribute\_name*

For example:

```
asadmin> get server1.jms-resource.myjms.*
```

Gets all the attributes for the JMS destination resource called `myjms`. For example:

```
server1.jms-resource.myjms.resType = javax.jms.Topic
server1.jms-resource.myjms.enabled = true
server1.jms-resource.myjms.name = myjms
server1.jms-resource.myjms.description = null
```

To get the value for a single attribute, for example `resType`:

```
asadmin> get server1.jms-resource.myjms.resType
```

To set an attribute, for example `description`:

```
asadmin> set server1.jms-resource.myjms.description=mydescription
```

This example sets the `description` to `mydescription`.

## Getting and Setting Multiple Values Examples

You can get and set multiple values with the same command. To set two attributes at the same time, separate the attributes by spaces. For example:

```
set server1.appReloadPollInterval=20 server1.mime.mime1.file=mime.types
```

Also, you can use the environment variable `AS_ADMIN_PREFIX` to set a prefix to use for a number of `get` and `set` commands.

## Monitoring Using `get` and `set` Commands

You can also use the `get` and `set` commands to monitor a running server. A `list` command also exists for monitoring. You can set an option, `monitor`, to `true` or `false`. If set to `true`, you monitor the attributes specified. For more information on using the command-line interface to monitor the Sun Java System Application Server, see [“Extracting Monitoring Data Using the CLI” on page 135](#).

## Using Help

Help for every `asadmin` command is available from the command prompt by typing `-h` or `--help`. For example, for help on `asadmin`, type:

```
asadmin --help
```

You see a list of all `asadmin` commands.

To get a help for a specific `asadmin` command, type:

```
asadmin command -h
```

or

```
asadmin command --help
```

The help contains a synopsis, a description of the command, syntax information, examples, and a list of related commands.

Please note that if you use `-h` or `--help` anywhere in a command, you will get help for the command. The command will not execute.

In a UNIX environment, you can also access the command-line help pages as manpages. You must first add *install\_dir*/man to your MANPATH environment variable. After setting the variable, you can access manpages for Sun Java System Application Server utilities by typing `man command_name` on the command line. For example:

```
man asadmin
```

If you have previously installed a different version of the Sun Java System Application Server (for example, Sun ONE Application Server 7, Enterprise Edition), make sure that your MANPATH environment variable points to your current installation directory.

## Viewing Output and Errors

When a command executes successfully, you see a message informing you of what was done. If the command fails, you see an error message.

This section contains the following topics:

- [Viewing the Exit Status](#)
- [Viewing Usage](#)

### Viewing the Exit Status

In addition to an error message, `asadmin` commands always exit with an exit status. The exit status is 0 if the command is successful and 1 if the command fails.

#### *Exit Status on UNIX*

You can check for the exit status at the command prompt by typing `echo $?`.

You can also use the exit codes in scripting; for example, the following Korn shell script uses the exit status to indicate whether the `list-instances` command succeeded or failed:

```
#!/bin/ksh
asadmin list-instances
if [[ $? = 0 ]]
then
    echo "success"
else
    echo "error"
fi
```



### *Exit Status on Windows*

On Windows you can check the exit status in your .bat scripts. For example, the following two scripts show a successful script and the output it returns, and an unsuccessful script and the output it returns:

#### **Success Condition**

```
myscript.bat
-----
echo off
echo Processing Command
call asadmin list-instances --domain domain1
if not %errorlevel% EQU 0 goto end
echo Command Successful
goto program-end
:end
echo Command Failed
:program-end
```

#### **Output:**

```
Processing Command
admin-server <not running>
server1 <not running>
Command Successful
```

#### **Error Condition**

```
myscript.bat
-----
echo off
echo Processing Command
call asadmin list-instances
if not %errorlevel% EQU 0 goto end
echo Command Successful
goto program-end
:end
echo Command Failed
:program-end
```

#### **Output:**

```
Processing Command
No default domain. Need to enter a domain.
Command Failed
```

## Viewing Usage

If you type the command without arguments, you get an error message that includes the syntax for the command. For example:

```
asadmin> create-instance

Invalid number of operands received

USAGE: create-instance [--user admin_user] [--password admin_password]
[--host localhost] [--port 4848] [--sysuser sys_user] [--domain
domain_name] [--local=false] [--passwordfile file_name] [secure | -s]
--instanceport instanceport instancename
```

## Security Considerations

When you run the command line interface from a command line, you must supply your password with all commands. If you are running in multimode, you must supply your password initially when you set up the environment. If you exit multimode, when you start multimode again you must set up your environment again, including your password. You set passwords using the environment commands. For more information see [“Using the Environment Commands” on page 565](#).

You can also set up a password file so you do not need to type your passwords at the command line. For more information, see [“Using the Password File Option” on page 567](#).

Without the authentication information of a valid username and password, the commands will not execute.

The command line interface has the security measures you’ve set up for your Sun Java System Application Server. For more information regarding security in the Sun Java System Application Server, see the *Sun Java System Application Server Administrator’s Security Guide*.

## Concurrent Access Considerations

It is possible that more than one person could attempt to configure a server concurrently using the command line interface and/or the Administration interface. If that happens, the second configuration request is queued until the first one completes. If the request is queued for too long, it times out.

For some commands, the changes don't take effect until you use the `reconfig` command. That means more than one person could edit an attribute before the changes are applied to the server. For more information on `reconfig`, see [“Starting and Stopping an Application Server Instance” on page 73](#).

## Command Reference

This section contains the following topics:

- [List of Commands](#)
- [List of Dotted Names and Attributes](#)
- [Long and Short Option Formats, Default Values, and Environment Variable Equivalents](#)

### List of Commands

The following table shows all the `asadmin` commands and their purpose. For more information on a command's syntax and usage, see the online help.

The commands marked *(Enterprise Edition)* are available in the Enterprise Edition release of the Sun Java System Application Server. All the other commands are part of both the Standard and Enterprise editions of the Sun Java System Application Server.

The left column shows the command name, and the right column shows its use.

**Table A-4** `asadmin` Commands

Command	Use
<code>add-resources</code>	Adds one or more resources of type jdbc, jms, or javamail.
<code>add-iiop-cluster-endpoint</code>	Adds an IIOP endpoint to the IIOP cluster. <i>(Enterprise Edition)</i>
<code>clear</code>	Reinitializes all the data space on all nodes.
<code>clearhistory</code>	Clears the history files on the database.
<code>clear-session-store</code>	Clears the sessions for all application server instances from the HADB database. <i>(Enterprise Edition)</i>
<code>configure-session-persistence</code>	Configures session persistence options for an application server instance. <i>(Enterprise Edition)</i>

**Table A-4** asadmin Commands (*Continued*)

Command	Use
create-acl	Creates an ACL (access control list).
create-authdb	Creates an authentication database.
create-auth-realm	Creates an authentication realm.
create-custom-resource	Creates a custom resource.
create-domain	Creates a domain.
create-file-user	Creates a file realm user in the keyfile.
create-http-listener	Creates an HTTP listener.
create-http-qos	Creates HTTP quality of service settings for the application server instance or virtual server.
create-iiop-listener	Creates an IIOP listener.
create-instance	Creates an application server instance.
create-javamail-resource	Creates a Java Mail resource.
create-jdbc-connection-pool	Creates a JDBC connection pool.
create-jdbc-resource	Creates a JDBC resource.
create-jmsdest	Creates a JMS (Java Message Service) destination.
create-jms-resource	Creates a JMS resource.
create-jndi-resource	Creates a JNDI resource.
create-jvm-options	Creates JVM options in java-config or profiler elements.
create-lifecycle-module	Creates a lifecycle module.
create-mime	Creates a MIME types file.
create-persistence-resource	Creates a persistence manager factory resource.
create-profiler	Creates a profiler for the JVM.
create-ssl	Creates SSL settings for an HTTP listener, IIOP listener, or IIOP service.
create-session-store	Creates a session store for a cluster to store the session information. ( <i>Enterprise Edition</i> )
create-virtual-server	Creates a virtual server.
delete-acl	Deletes an ACL.
delete-authdb	Deletes an authentication database.
delete-auth-realm	Deletes an authentication realm.
delete-custom-resource	Deletes a custom resource.

**Table A-4** asadmin Commands (*Continued*)

Command	Use
delete-domain	Deletes a domain. This command can only be executed locally.
delete-file-user	Deletes a file realm user from the keyfile.
delete-http-listener	Deletes an HTTP listener.
delete-http-qos	Deletes HTTP quality of service settings for the application server instance or virtual server.
delete-iiop-cluster-endpoint	Deletes IIOP endpoint from the IIOP cluster configuration. ( <i>Enterprise Edition</i> )
delete-iiop-listener	Deletes an IIOP listener
delete-instance	Deletes an application server instance.
delete-javamail-resource	Deletes a Java Mail resource.
delete-jdbc-connection-pool	Deletes a JDBC connection pool.
delete-jdbc-resource	Deletes a JDBC resource.
delete-jmsdest	Deletes a JMS destination.
delete-jms-resource	Deletes a JMS resource.
delete-jndi-resource	Deletes a JNDI resource.
delete-jvm-options	Deletes JVM options in java-config or profiler elements.
delete-lifecycle-module	Deletes a lifecycle module.
delete-mime	Deletes a MIME types file.
delete-persistence-resource	Deletes a persistence manager factory resource.
delete-profiler	Deletes a JVM profiler.
delete-ssl	Deletes SSL settings for an HTTP listener, IIOP listener, or IIOP service.
delete-virtual server	Deletes a virtual server.
deploy	Deploys an EJB, WEB, connector, appclient, or application component to the application server instance.
deploydir	Deploys an EJB, WEB, connector, appclient, or application component that is in the directory to the application server instance.
disable	Disables a deployed component in the application server instance.
display-license	Displays license information. This command can only be executed locally.

**Table A-4** `asadmin` Commands (*Continued*)

Command	Use
<code>enable</code>	Enables (allows to run) a deployed component in the application server instance.
<code>export</code>	Exports the value of an <code>asadmin</code> environment variable so that it can be used by the subsequent <code>asadmin</code> commands.
<code>get</code>	Gets the value of an attribute.
<code>help</code>	Displays help (description, usage, syntax, examples) for a given command, or general help for <code>asadmin</code> .
<code>install-license</code>	Installs the license file. This command can only be executed locally.
<code>jms-ping</code>	Pings the JMS provider to see if it is running.
<code>list</code>	Lists the configurable elements.
<code>list-acls</code>	Lists ACLs for an application server instance.
<code>list-authdbs</code>	Lists authentication databases.
<code>list-auth-realms</code>	Lists authentication realms.
<code>list-components</code>	Lists the deployed components for a server instance.
<code>list-custom-resources</code>	Lists custom resources in a server instance
<code>list-domains</code>	Lists domains.
<code>list-file-users</code>	Lists all the file realm users in a server instance.
<code>list-file-groups</code>	Lists all the groups for a specified file realm user. If you do not specify a user, lists all groups for a server instance.
<code>list-http-listeners</code>	Lists HTTP listeners for a server instance.
<code>list-instances</code>	Lists application server instances in the domain.
<code>list-iiop-cluster-config</code>	Lists the IIOP cluster configuration. ( <i>Enterprise Edition</i> )
<code>list-iiop-listeners</code>	Lists IIOP listeners for a server instance.
<code>list-javamail-resources</code>	Lists Java Mail resources for a server instance.
<code>list-jdbc-connection-pools</code>	Lists JDBC connection pools for a server instance.
<code>list-jdbc-resources</code>	Lists JDBC resources for a server instance.
<code>list-jmsdest</code>	Lists JMS destinations for a server instance.
<code>list-jms-resources</code>	Lists JMS resources for a server instance
<code>list-jndi-resources</code>	Lists JNDI resources for a server instance.
<code>list-lifecycle-modules</code>	Lists lifecycle modules for a server instance.
<code>list-mimes</code>	Lists MIME types files for a server instance.

**Table A-4** asadmin Commands (*Continued*)

Command	Use
list-persistence-resources	Lists persistence manager factory resources for a server instance.
list-profilers	Lists JVM profilers for a server instance.
list-sub-components	Lists one or more EJBs or Servlets in a deployed module or in a module of the deployed application.
list-virtual-servers	Lists virtual servers for a server instance.
multimode	Allows you to execute multiple command while retaining your environment settings and remaining within asadmin.
reconfig	Applies change to the server. Most changes do not take effect until they are applied.
restart-instance	Restarts the server instance.
set	Sets the value of an attribute.
show-component-status	Shows the status of a deployed component.
show-instance-status	Shows the status of a server instance (that is, whether it is running or not).
shutdown	Shuts down the Administration Server.
start-appserv	Starts the Administration Server and all the server instances. This command can only be executed locally.
start-domain	Starts all instances in the domain. This command can only be executed locally.
start-instance	Starts the server instance.
stop-appserv	Stops the Administration Server and all the server instances. This command can only be executed locally.
stop-domain	Stops all instances in the domain.
stop-instance	Stops the server instance.
undeploy	Removes the deployed component from the server instance.
unset	Unsets the exported environment variables for asadmin.
update-file-user	Updates an existing file realm user.
version	Displays version information for the Sun Java System Application Server.

# List of Dotted Names and Attributes

When you use the `get` and `set` commands to get and set attributes, you need to know the names `asadmin` uses for the services, resources, and so forth so you can use the name to get the attributes for that particular object.

Because the syntax for using these names involves separating names between periods, these names are called dotted names.

## Dotted Names Used in asadmin

The following tables list the names used to configure items using the `asadmin`. They are broken out into the following categories:

- [Service Names](#)
- [Resource Names](#)
- [Application Names](#)
- [Other Names](#)

### Service Names

The following table shows the service names to use to get and set attributes for the services:

**Table A-5** Service Names for Command-Line Interface

Service	Dotted Name
JMS service configuration	<a href="#">jms-service</a>
Transaction service configuration	<a href="#">transaction-service</a>
MDB container configuration	<a href="#">mdb-container</a>
EJB container configuration	<a href="#">ejb-container</a>
Web container configuration	<a href="#">web-container</a>
JVM configuration	<a href="#">java-config</a>
ORB configuration	<a href="#">orb</a> or <a href="#">iiop-service</a>



**Table A-5** Service Names for Command-Line Interface *(Continued)*

Service	Dotted Name
ORB listener configuration	<code>orblister</code> or <code>iiop-listener</code>  Note that <code>orblister</code> and <code>iiop-listener</code> are not valid names as is; both require the names of the listener to follow. For example:  <code>orblister.listener_name</code> or <code>iiop-listener.listener_name</code>
Log configuration	<code>log-service</code>
Security configuration	<code>security-service</code>
HTTP configuration	<code>http-service</code>

## Resource Names

The following table shows resource names to use to get and set attributes for the resources. Note that these names are not valid by themselves; they require the name of a resource to follow the resource name.

**Table A-6** Resource Names for Command-Line Interface

Resource	Dotted Name
JDBC resource configuration	<code>jdbc-resource</code>
JNDI resource configuration	<code>jndi-resource</code>
JDBC connection pool resource configuration	<code>jdbc-connection-pool</code>
Custom resource configuration	<code>custom-resource</code>
JMS resource configuration	<code>jms-resource</code>
Persistence manager factory resource configuration	<code>persistence-manager-factory-resource</code>
Java mail resource configuration	<code>mail-resource</code>

## Application Names

The following table shows the dotted names to use to get and set attributes application-related configuration. Note that these names are not valid by themselves; they require the name of the application to follow.

**Table A-7** Application Names for Command-Line Interface

Application Component	Dotted Name
Application configuration	<a href="#">application</a>
EJB module configuration	<a href="#">ejb-module</a>
Web module configuration	<a href="#">web-module</a>
Connector module configuration	<a href="#">connector-module</a>

## Other Names

The following table shows the dotted names of other items you can configure using `get` and `set`. Note that these names are not valid by themselves; they require the name of the application to follow; for example, `http-listener.listener_name`, `lifecycle-module.module-name`, etc.

**Table A-8** Other Item Names for Command-Line Interface

Item	Dotted Name
HTTP listener	<a href="#">http-listener</a> or <a href="#">http-server.http-listener</a>
MIME types file	<a href="#">mime</a>
ACL	<a href="#">acl</a>
Virtual server	<a href="#">virtual-server</a>
Authentication databases	<a href="#">auth-db</a>
Security realms	<a href="#">authrealm</a>
Lifecycle module	<a href="#">lifecycle-module</a>
Profiler configuration	<a href="#">profiler</a>
Server configuration	<a href="#">server configuration (name of server instance)</a>

## Attributes

The following sections show the attributes for each named item listed above, and provide usage examples. Note that some attributes are read-only (that is, they can only be used with the `get` command, not the `set` command).

---

**NOTE**

The examples in this section assume the user, password, host and port are defined in the environment variables, and don't list those options in the syntax.

---

jms-service

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-9** JMS Service Attributes

server.xml Name	asadmin Name
port	port
admin-username	adminUserName
admin-password	adminPassword
log-level	logLevel
enabled	enabled
init-timeout-in-seconds	initTimeoutInSeconds
start-args	startArgs

To get all the attributes from an instance (server1):

```
asadmin> get server1.jms-service.*
```

To get an attribute called `adminPassword`:

```
asadmin> get server1.jms-service.adminPassword
```

To set an attribute called `adminPassword` to a value of `admin`:

```
asadmin> set server1.jms-service.adminPassword=admin
```

transaction-service

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-10** Transaction Service Attributes

server.xml Name	asadmin Name
automatic-recovery	automaticTransactionRecovery
timeout-in-seconds	transactionRecoveryTimeout
tx-log-dir	transactionLogFile
heuristic-decision	heuristicDecision
keypoint-interval	keypointInterval
log-level	logLevel
monitoring-enabled	monitoringEnabled

To get all the attributes from an instance (server1):

```
asadmin> get server1.transaction-service.*
```

To get an attribute called transactionRecoveryTimeout:

```
asadmin> get server1.transaction-service.transactionRecoveryTimeout
```

To set an attribute called transactionRecoveryTimeout to a value of 49:

```
asadmin> set server1.transaction-service.transactionRecoveryTimeout=49
```

### mdb-container

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-11** MDB Container Attributes

server.xml Name	asadmin Name
steady-pool-size	steadyPoolSize
pool-resize-quantity	poolResizeQuantity
max-pool-size	maxPoolSize
idle-timeout-in-seconds	idleInPoolTimeoutInSeconds
log-level	logLevel
monitoring-enabled	monitoringEnabled

To get all the attributes from an instance (server1):

```
asadmin> get server1.mdb-container.*
```

To get an attribute called `steadyPoolSize`:

```
asadmin> get server1.mdb-container.steadyPoolSize
```

To set an attribute called `steadyPoolSize` to a value of 10:

```
asadmin> set server1.mdb-container.steadyPoolSize=10
```

## ejb-container

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-12** EJB Container Attributes

server.xml Name	asadmin Name
steady-pool-size	steadyPoolSize
pool-resize-quantity	poolResizeQuantity
max-pool-size	maxPoolSize
cache-resize-quantity	cacheResizeQuantity
max-cache-size	maxCacheSize
pool-idle-timeout-in-seconds	idleInPoolTimeoutInSeconds
cache-idle-timeout-in-seconds	idleInCacheTimeoutInSeconds
removal-timeout-in-seconds	removalTimeoutInSeconds
victim-selection-policy	victimSelectionPolicy
commit-option	commitOption
log-level	logLevel
monitoring-enabled	monitoringEnabled

To get all the attributes from an instance (server1):

```
asadmin> get server1.ejb-container.*
```

To get an attribute called `maxPoolSize`:

```
asadmin> get server1.ejb-container.maxPoolSize
```

To set an attribute called `maxPoolSize` to a value of 12:

```
asadmin> set server1.ejb-container.maxPoolSize=12
```

web-container

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-13** Web Container Attributes

server.xml Name	asadmin Name
log-level	logLevel
monitoring-enabled	monitoringEnabled (not used)

To get all the attributes from an instance (server1):

```
asadmin> get server1.web-container.*
```

To get an attribute called `logLevel`:

```
asadmin> get server1.web-container.logLevel
```

To set an attribute called `monitoringEnabled` to be `WARNING`:

```
asadmin> set server1.web-container.logLevel=WARNING
```

java-config

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-14** JVM Attributes

server.xml Name	asadmin Name
java-home	javahome
debug-enabled	debugEnabled
debug-options	debugOptions
javac-options	javacoptions
rmic-options	rmicoptions
classpath-prefix	classpathprefix
server-classpath	serverClasspath
classpath-suffix	classpathsuffix

**Table A-14** JVM Attributes (*Continued*)

<b>server.xml Name</b>	<b>asadmin Name</b>
native-library-path-prefix	libpathprefix
native-library-path-suffix	libpathsuffix
env-classpath-ignored	envpathignore

To get all the attributes from an instance (server1):

```
asadmin> get server1.java-config.*
```

To get an attribute called `classpathprefix`:

```
asadmin> get server1.java-config.classpathprefix
```

To set an attribute called `classpathprefix` to a value of `com.sun`:

```
asadmin> set server1.java-config.classpathprefix=com.sun
```

## orb or iiop-service

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-15** ORB/IIOP Service Attributes

<b>server.xml Name</b>	<b>asadmin Name</b>
message-fragment-size	msgSize
steady-thread-pool-size	minThreads
max-thread-pool-size	maxThreads
max-connections	maxConnections
idle-thread-timeout-in-seconds	idleThreadTimeout
log-level	log
monitoring-enabled	monitor
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls

**Table A-15** ORB/IIOP Service Attributes (*Continued*)

server.xml Name	asadmin Name
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

To get all the attributes from an instance (server1):

```
asadmin> get server1.orb.*
```

or

```
asadmin> get server1.iiop-service.*
```

To get an attribute called `msgSize`:

```
asadmin> get server1.orb.msgSize
```

or

```
asadmin> get server1.iiop-service.msgSize
```

To set an attribute called `idleThreadTimeout` to 300:

```
asadmin> set server1.orb.idleThreadTimeout=300
```

or

```
asadmin> set server1.iiop-service.idleThreadTimeout=300
```

**orblister or iiop-listener**

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-16** IIOP Listener Attributes

server.xml Name	asadmin Name
id	id
address	address
port	port
enabled	enabled
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers



**Table A-16** IIOP Listener Attributes (*Continued*)

server.xml Name	asadmin Name
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

To get all the attributes from an instance (server1):

```
asadmin> get server1.orblistener.orb_listener_id.*
```

or

```
asadmin> get server1.iiop-listener.orb_listener_id.*
```

To get an attribute called port:

```
asadmin> get server1.orblistener.orb_listener_id.port
```

or

```
asadmin> get server1.iiop-listener.orb_listener_id.port
```

To set an attribute called address to bluestar:

```
asadmin> set server1.orblistener.orb_listener_id.address=bluestar
```

or

```
asadmin> set server1.iiop-listener.orb_listener_id.address=bluestar
```

## log-service

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-17** Log Configuration Attributes

server.xml Name	asadmin Name
file	file
level	level
log-stdout	stdout
log-stderr	stderr

**Table A-17** Log Configuration Attributes (*Continued*)

server.xml Name	asadmin Name
echo-log-messages-to-stderr	echoToStderr
create-console	createConsole
log-virtual-server-id	LogVirtualServerId
use-system-logging	useSystemLogging

To get all the attributes from an instance (server1):

```
asadmin> get server1.log-service.*
```

To get an attribute called `level`:

```
asadmin> get server1.log-service.level
```

To set an attribute called `echoToStderr` to `true`:

```
asadmin> set server1.log-service.echoToStderr=true
```

### security-service

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-18** Security Realm Configuration Attributes

server.xml Name	asadmin Name
default-realm	defaultRealm
default-principal	defaultPrincipal
default-principal-password	defaultPrincipalPassword
anonymous-role	anonymousRole
audit-enabled	auditEnabled
log-level	logLevel

To get all the attributes from an instance (server1):

```
asadmin> get server1.security-service.*
```

To get an attribute called `anonymousRole`:

```
asadmin> get server1.security-service.anonymousRole
```

To set an attribute called `encryptPasswords` to `true`:

```
asadmin> set server1.security-service.auditEnabled=true
```

http-service

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-19** HTTP Service Attributes

server.xml Name	asadmin Name
qos-metrics-interval-in-seconds	qos-metrics-interval-in-seconds
qos-recompute-time-interval-in-millis	qos-recompute-time-interval-in-millis
qos-enabled	qos-enabled
bandwidth-limit	bandwidthLimit
enforce-bandwidth-limit	enforceBandwidthLimit
connection-limit	connectionLimit
enforce-connection-limit	enforceConnectionLimit

To get all the attributes from an instance (`server1`):

```
asadmin> get server1.http-service.*
```

To get an attribute called `bandwidthLimit`:

```
asadmin> get server1.http-service.bandwidthLimit
```

To set an attribute called `qos-enabled` to `true`:

```
asadmin> set server1.http-service.qos-enabled=true
```

jdbc-resource

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-20** JDBC Resource Attributes

server.xml Name	asadmin Name
jndi-name	name
pool-name	pool
enabled	enabled
description	description

To get all the attributes from an instance (server1):

```
asadmin> get server1.jdbc-resource.jdbc_resource_name.*
```

To get an attribute called pool:

```
asadmin> get server1.jdbc-resource.jdbc_resource_name.pool
```

To set an attribute called enabled to true:

```
asadmin> set server1.jdbc-resource.jdbc_resource_name.enabled=true
```

### jndi-resource

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-21** JNDI Resource Attributes

server.xml Name	asadmin Name
jndi-name	name
jndi-lookup-name	LookupName
res-type	resType
factory-class	factory
enabled	enabled
description	description

To get all the attributes from an instance (server1):

```
asadmin> get server1.jndi-resource.jndi_name.*
```

To get an attribute called `factory`:

```
asadmin> get server1.jndi-resource.jndi_name.factory
```

To set an attribute called `factory` to `com.sun`:

```
asadmin> set server1.jndi-resource.jndi_name.factory=com.sun
```

### jdbc-connection-pool

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-22** JDBC Connection Pool Attributes

server.xml Name	asadmin Name
name	name
datasource-classname	dsClassName
res-type	resType
description	description
steady-pool-size	steadyPoolSize
max-pool-size	maxPoolSize
max-wait-time-in-millis	maxWaitTime
pool-resize-quantity	resizeValue
idle-timeout-in-seconds	idleTimeout
transaction-isolation-level	transactionIsolationLevel
is-isolation-level-guaranteed	isIsolationLevelGuaranteed
connection-validation-method	validationMethod
is-connection-validation-required	isValidationRequired
fail-all-connections	failAll
validation-table-name	validationTable

To get all the attributes from an instance (`server1`):

```
asadmin> get server1.jdbc-connection-pool.pool_name.*
```

To get an attribute called `dsClassName`:

```
asadmin> get server1.jdbc-connection-pool.pool_name.dsClassName
```

To set an attribute called `resizeValue` to 2:

```
asadmin> set server1.jdbc-connection-pool.pool_name.resizeValue=2
```

custom-resource

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-23** Custom Resource Attributes

server.xml Name	asadmin Name
jndi-name	name
res-type	resType
factory-class	factory
enabled	enabled
description	description

To get all the attributes from an instance (`server1`):

```
asadmin> get server1.custom-resource.jndi_name.*
```

To get an attribute called `factory`:

```
asadmin> get server1.custom-resource.jndi_name.factory
```

To set an attribute called `factory`:

```
asadmin> set server1.custom-resource.jndi_name.factory=myclass
```

jms-resource

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-24** JMS Resource Attributes

server.xml Name	asadmin Name
jndi-name	name
res-type	resType
enabled	enabled
description	description

To get all the attributes from an instance (server1):

```
asadmin> get server1.jms-resource.jms_resource_name.*
```

To get an attribute called `res-type`:

```
asadmin> get server1.jms-resource.jms_resource_name.resType
```

To set an attribute called `enabled` to `true`:

```
asadmin> set server1.jms-resource.jms_resource_name.enabled=true
```

**persistence-manager-factory-resource**

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-25** Persistence Manager Factory Resource Attributes

server.xml Name	asadmin Name
jndi-name	jndiName
jdbc-resource-jndi-name	JdbcResourceJndiName
factory-class	factoryClass
enabled	enabled
description	description

To get all the attributes from an instance (server1):

```
asadmin> get server1.persistence-manager-factory-resource.jndi_name
```

To get an attribute called `factoryClass`:

```
asadmin> get  
server1.persistence-manager-factory-resource.jndi_name.factoryClass
```

To set an attribute called `enabled` to `true`:

```
asadmin> set  
server1.persistence-manager-factory-resource.jndi_name.enabled=true
```

**mail-resource**

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-26** Java Mail Resource Attributes

server.xml Name	asadmin Name
jndi-name	name
enabled	enabled
store-protocol	storeProtocol
store-protocol-class	storeProtocolClass
transport-protocol	transportProtocol
transport-protocol-class	transportProtocolClass
host	host
user	user
from	from
debug	debug
description	description

To get all the attributes from an instance (server1):

```
asadmin> get server1.mail-resource.jndi_name.*
```

To get an attribute called `host`:

```
asadmin> get server1.mail-resource.jndi_name.host
```

To set an attribute called `enabled` to `true`:

```
asadmin> set server1.mail-resource.jndi_name.enabled=true
```

## application

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-27** Application Attributes

server.xml Name	asadmin Name
name	name
location	location
virtual-servers	virtualServers



**Table A-27** Application Attributes (*Continued*)

<b>server.xml Name</b>	<b>asadmin Name</b>
description	description
enabled	enabled

To get all the attributes from an instance (server1):

```
asadmin> get server1.application.application_name.*
```

To get an attribute called `location` in an application:

```
asadmin> get server1.application.application_name.location
```

To set an attribute called `location`:

```
asadmin> set server1.application.application_name.location=
"/export/home/as7se/as1/repository/applications/ASConverter"
```

## ejb-module

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-28** EJB Module Attributes

<b>server.xml Name</b>	<b>asadmin Name</b>
name	name
location	location
description	description
enabled	enabled

To get all the attributes of stand-alone EJB module in an instance (server1):

```
asadmin> get server1.ejb-module.ejb_jar_name.*
```

To get all the attributes of an EJB module in an application for an instance (server1):

```
asadmin> get server1.j2ee-application.application_name.ejb-module.ejb_jar_name.*
```

or

```
asadmin> get server1.application.application_name.ejb-module.ejb_jar_name.*
```

To get an attribute called `location` from a stand-alone EJB module:

```
asadmin> get server1.ejb-module.ejb_jar_name.location
```

To get an attribute called `location` from an EJB module in an application:

```
asadmin> get server1.j2ee-application.application_name.ejb-module.ejb_jar_name.location
```

or

```
asadmin> get server1.application.application_name.ejb-module.ejb_jar_name.location
```

To set an attribute called `location` in the stand-alone EJB module:

```
asadmin> set
server1.ejb-module.ejb_jar_name.location="/export/home/as7se/as1/repository/modules/ejb_jar_name"
```

To set an attribute called `location` in the EJB module bundled into an application:

```
asadmin> set server1.j2ee-application.application_name.ejb-module.ejb_jar_name.location="/export/home/as7se/as1/repository/modules/ejb_jar_name"
```

or

```
asadmin>set
server1.application.application_name.ejb-module.ejb_jar_name.location="/export/home/as7se/as1/repository/modules/ejb_jar_name"
```

web-module

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-29** WEB Module Attributes

server.xml Name	asadmin Name
name	name
location	location
context-root	contextRoot
virtual-servers	virtualServers
description	description
enabled	enabled

To get all the attributes of stand-alone web module in an instance (server1):

```
asadmin> get server1.web-module.web_war_name.*
```

To get all the attributes of a web module in an application for an instance (server1):

```
asadmin> get server1.web-module.application_name.web_war_name.*
```

To get an attribute called `location` from stand-alone web module:

```
asadmin> get server1.web-module.web_war_name.location
```

To get an attribute called `location` from the web module in an application:

```
asadmin> get server1.web-module.application_name.web_war_name.location
```

To set an attribute called `location` in the standalone web module:

```
asadmin> set server1.web-module.war-ic.location=  
"/export/home/as7se/as1/repository/modules/web_war_name"
```

To set an attribute called `location` in the web module bundled into an application:

```
asadmin> set server1.web-module.application_name.web_war_name.location=  
"/export/home/as7se/as1/repository/modules/web_war_name"
```

connector-module

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-30** Connector Module Attributes

server.xml Name	asadmin Name
name	name
location	location
description	description
enabled	enabled

To get all the attributes of standalone connector module in an instance (server1):

```
asadmin> get server1.connector-module.connector_rar_name.*
```

To get an attribute called `location` from standalone connector module:

```
asadmin> get server1.connector-module.connector_rar_name.location
```

To set an attribute called `location` in the standalone connector module:

```
asadmin> set server1.connector-module.connector_rar_name.location=
"/export/home/as7se/as1/repository/modules/connector_rar_name"
```

http-listener or http-server.http-listener

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-31** HTTP Listener Attributes

server.xml Name	asadmin Name
id	id
address	address
port	port
family	family
acceptor-threads	acceptorThreads
blocking-enabled	blockingEnabled
security-enabled	securityEnabled
default-virtual-server	defaultVirtualServer
server-name	serverName
enabled	enabled
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

To get all the attributes from an instance (server1):

```
asadmin> get server1.http-listener.http_listener_name.*
```

or

```
asadmin> get server1.http-server.http-listener.http_listener_name.*
```

To get an attribute called `factory`:

```
asadmin> get server1.http-listener.http_listener_name.address
```

or

```
asadmin> get server1.http-server.http-listener.http_listener_name.address
```

To set an attribute called `address` to the IP address `0.0.0.0`:

```
asadmin> set server1.http-listener.http_listener_name.address=0.0.0.0
```

or

```
asadmin> set  
server1.http-server.http-listener.http_listener_name.address=0.0.0.0
```

mime

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-32** MIME Types Attributes

server.xml Name	asadmin Name
id	id
file	file

To get all the attributes from an instance (`server1`):

```
asadmin> get server1.mime.mime_name.*
```

To get an attribute called `file`:

```
asadmin> get server1.mime.mime_name.file
```

To set an attribute called `file` to `mime.types`:

```
asadmin> set server1.mime.mime_name.file=mime.types
```

acl

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-33** ACL Attributes

server.xml Name	asadmin Name
id	id
file	file

To get all the attributes from an instance (server1):

```
asadmin> get server1.acl.acl_name.*
```

To get an attribute called file:

```
asadmin> get server1.acl.acl_name.file
```

To set an attribute called file:

```
asadmin> set server1.acl.acl_name.file=com/as1.acl
```

virtual-server

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-34** Virtual Server Attributes

server.xml Name	asadmin Name
id	id
http-listeners	httpListeners
config-file	configFile
default-object	defaultObject
accept-language	acceptLanguage
log-file	logFile
default-web-module	defaultWebModule
hosts	hosts
mime	mime
state	state
acls	acls
bandwidth-limit	bandwidthLimit
enforce-bandwidth-limit	enforceBandwidthLimit

**Table A-34** Virtual Server Attributes (*Continued*)

server.xml Name	asadmin Name
connection-limit	connectionLimit
enforce-connection-limit	enforceConnectionLimit
property name="dir" value=	property.dir
property name="nice" value=	property.nice
property name="user" value=	property.user
property name="group" value=	property.group
property name="chroot" value=	property.chroot
property name="docroot" value=	property.docroot
property name="accesslog" value=	property.accesslog

To get all the attributes from an instance (server1):

```
asadmin> get instance_name.virtual-server.vserver_id.*
```

For example:

```
asadmin> get server1.virtual-server.server1.*
```

To get an attribute called `httpListeners` for virtual server `server1`:

```
asadmin> get server1.virtual-server.server1.httpListeners
```

To set an attribute called `acceptLanguage` to `false`:

```
asadmin> set server1.virtual-acceptLanguage=false
```

## auth-db

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-35** Authentication Database Attributes

server.xml Name	asadmin Name
id	id
database	database
basedn	basedn
certmaps	certmaps

To get all the attributes from an instance:

```
asadmin> get instanceName.virtual-server.vserver_id.auth-db.authdb_id.*
```

For example, for the instance server1, virtual server server1:

```
asadmin> get server1.virtual-server.server1.auth-db.authdb_id.*
```

To get an attribute called database:

```
asadmin> get server1.virtual-server.server1.auth-db.authdb_id.database
```

To set an attribute called database:

```
asadmin> set server1.virtual-server.server1.auth-db.authdb_id.database=Oracle
```

authrealm

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-36** Authorization Realm Attributes

server.xml Name	asadmin Name
name	name
classname	classname

To get all the attributes from an instance (server1):

```
asadmin> get server1.authrealm.authrealm_id.*
```

To get an attribute called classname:

```
asadmin> get server1.authrealm.authrealm_id.classname
```

To set an attribute called classname:

```
asadmin> set
server1.authrealm.authrealm_id.classname=com.sun.as.security.auth.realm.shared
password.SharedPasswordRealm
```

lifecycle-module

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.



**Table A-37** LifeCycle Module Attributes

server.xml Name	asadmin Name
name	name
enabled	enabled
class-name	className
classpath	classPath
load-order	loadOrder
is-failure-fatal	isFailureFatal
description	description

To get all the attributes from an instance (server1):

```
asadmin> get server1.lifecycle-module.lifecycle_module_id.*
```

To get an attribute called `className` for a lifecycle module:

```
asadmin> get server1.lifecycle-module.lifecycle_module_id.className
```

To set an attribute called `className`:

```
asadmin> set
server1.lifecycle-module.lifecycle_module_id.className=com.lifecycle_module_id.lifecycle
e
```

## profiler

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-38** JVM Profiler Configuration Attributes

server.xml Name	asadmin Name
name	name
classpath	classPath
native-library-path	nativeLibraryPath
enabled	enabled

To get all the attributes from an instance (server1):

```
asadmin> get server1.profiler.*
```

To get an attribute called enabled:

```
asadmin> get server1.profiler.enabled
```

To set an attribute called enabled to false

```
asadmin> set server1.profiler.enabled=false
```

server configuration (name of server instance)

The following table shows the `server.xml` name for the attribute in the left column, and the name used by `asadmin` in the right column.

**Table A-39** Server Configuration Attributes

server.xml Name	asadmin Name
instance-name	name
locale	locale
log-root	logRoot
session-store	sessionStore
application-root	applicationRoot
dynamic-reload-enabled	appDynamicReloadEnabled
dynamic-reload-poll-interval-in-seconds	appReloadPollInterval

To get all the attributes from an instance (server1):

```
asadmin> get server1.*
```

To get an attribute called logRoot:

```
asadmin> get server1.logRoot
```

To set an attribute called logRoot:

```
asadmin> set server1.logRoot="/space/log"
```

# Long and Short Option Formats, Default Values, and Environment Variable Equivalents

The following table lists the long and short formats for command line options. If there is no short format listed, a short format of the option is not available.

**Table A-40** Short and Long Options, Default Values, and Environment Variables

Option Name	Long Format	Short Format	Default Value	Environment Variable
acceptlang	--acceptlang			AS_ADMIN_ACCEPT_
acceptorthreads	--acceptorthreads			AS_ADMIN_ACCEPTOR_THREADS
acls	--acls			AS_ADMIN_ACLS
address	--address			AS_ADMIN_ADDRESS
adminpassword	--adminpassword			AS_ADMIN_ADMINPASSWD
adminport	--adminport		4848	AS_ADMIN_ADMINPORT
adminuser	--adminuser			AS_ADMIN_ADMINUSER
basedn	--basedn			AS_ADMIN_BASEDN
blockingenabled	--blockingenabled			AS_ADMIN_BLOCKINGENABLED
bwlimit	--bwlimit			AS_ADMIN_BWLIMIT
certmaps	--certmaps			AS_ADMIN_CERTMAPS
certname	--certname			AS_ADMIN_CERTNAME
classname	--classname			AS_ADMIN_CLASSNAME
classpath	--classpath			AS_ADMIN_CLASSPATH
clientauthenabled	--clientauthenabled			AS_ADMIN_CLIENTAUTHENABLED
configfile	--configfile			AS_ADMIN_CONFIGFILE
connectionpoolid	--connectionpoolid			AS_ADMIN_CONNECTIONPOOLID
connlimit	--connlimit			AS_ADMIN_CONNLIMIT
contextroot	--contextroot			AS_ADMIN_CONTEXTROOT
database	--database			AS_ADMIN_DATABASE
debug	--debug		false	AS_ADMIN_DEBUG
defaultobj	--defaultobj			AS_ADMIN_DEFAULTOBJ
defaultwebmodule	--defaultwebmodule			AS_ADMIN_DEFAULTWEBMODULE
description	--description			AS_ADMIN_DESCRIPTION

**Table A-40** Short and Long Options, Default Values, and Environment Variables (*Continued*)

Option Name	Long Format	Short Format	Default Value	Environment Variable
discardmanualchanges	--discardmanualchanges	-d	false	AS_ADMIN_DISCARDMANUALCHANGES
echo	--echo	-e	false	AS_ADMIN_ECHO
enabled	--enabled			AS_ADMIN_ENABLED
enforcebwlimit	--enforcebwlimit			AS_ADMIN_ENFORCEBWLIMIT
enforceconnnlimit	--enforceconnnlimit			AS_ADMIN_ENFORCECONNNLIMIT
failconnection	--failconnection		false	AS_ADMIN_FAILCONNECTION
failurefatal	--failurefatal		false	AS_ADMIN_FAILUREFATAL
family	--family			AS_ADMIN_FAMILY
file	--file	-f		AS_ADMIN_FILE
force	--force	-F	true	AS_ADMIN_FORCE
help	--help	-h		AS_ADMIN_HELP
host	--host	-H		AS_ADMIN_HOST
hosts	--hosts			AS_ADMIN_HOSTS
httplistenerid	--httplistenerid			AS_ADMIN_HTTPLISTENERID
httplisteners	--httplisteners			AS_HTTP_LISTENERS
idletimeout	--idletimeout		300	AS_ADMIN_IDLETIMEOUT
instance	--instance	-i	server1	AS_ADMIN_INSTANCE
instanceport	--instanceport			AS_ADMIN_INSTANCEPORT
interactive	--interactive	-I	true	AS_AMDIN_INTERACTIVE
isconnectvalidaterequired	--isconnectvalidaterequired		false	AS_ADMIN_ISCONNECTVALIDATEREQUIRED
jdbcjndiname	--jdbcjndiname	-a		AS_ADMIN_JDBCJNDINAME
jndilookupname	--jndilookupname	-l		AS_ADMIN_JNDILOOKUPNAME
keepmanualchanges	--keepmanualchanges	-k	false	AS_ADMIN_KEEPMANUALCHANGES
loadorder	--loadorder			AS_ADMIN_LOADORDER
local	--local	-l	false	
logfile	--logfile			AS_ADMIN_LOGFILE
maxpoolsize	--maxpoolsize		32	AS_ADMIN_MAXPOOLSIZE

**Table A-40** Short and Long Options, Default Values, and Environment Variables (*Continued*)

Option Name	Long Format	Short Format	Default Value	Environment Variable
maxwait	--maxwait		6000	AS_ADMIN_MAXWAIT
mime	--mime			AS_ADMIN_MIME
mimefile	--mimefile			AS_ADMIN_MIMEFILE
monitor	--monitor	-m	false	AS_ADMIN_MONITOR
name	--name	-n		AS_ADMIN_NAME
nativelibpath	--nativelibpath			AS_ADMIN_NATIVELIBPATH
objtype	--objtype	-o		AS_ADMIN_OBJTYPE
password	--password	-w		AS_ADMIN_PASSWORD
poolresize	--poolresize		2	AS_ADMIN_POOLRESIZE
port	--port	-p	8000	AS_ADMIN_PORT
prefix	--prefix	-x		AS_ADMIN_PREFIX
printprompt	--printprompt	-P	true	AS_ADMIN_PROMPT
property	--property			AS_ADMIN_PROPERTY
securityenabled	--securityenabled			AS_ADMIN_SECURITYENABLED
servername	--servername			AS_ADMIN_SERVERNAME
ssl2ciphers	--ssl2ciphers			AS_ADMIN_SSL2CIPHERS
ssl2enabled	--ssl2enabled			AS_ADMIN_SSL2ENABLED
ssl3enabled	--ssl3enabled			AS_ADMIN_SSL3ENABLED
ssl3tlsciphers	--ssl3tlsciphers			AS_ADMIN_SSL3TLSCIPHERS
state	--state			AS_ADMIN_STATE
steadypoolsize	--steadypoolsize	8		AS_ADMIN_STEADYPOOLSIZE
storeprotocol	--storeprotocol			AS_ADMIN_STOREPROTOCOL
storeprotocolclass	--storeprotocolclass			AS_ADMIN_STOREPROTOCOLCLASS
tlsenabled	--tlsenabled			AS_ADMIN_TLSENABLED
tlsrollbackenabled	--tlsrollbackenabled			AS_ADMIN_TLSROLLBACKENABLED
transprotocol	--transprotocol		smtp	AS_ADMIN_TRANSPROTOCOL
type	--type			S_ADMIN_TRANSPROTOCOLCLASS
upload	--upload	-U	true	AS_ADMIN_TYPE
url	--url			AS_ADMIN_URL

**Table A-40** Short and Long Options, Default Values, and Environment Variables *(Continued)*

Option Name	Long Format	Short Format	Default Value	Environment Variable
user	--user	-u		AS_ADMIN_USER
validationmethod	--validationmethod		auto-commit	AS_ADMIN_VALIDATIONMETHOD
validationtable	--validationtable			AS_ADMIN_VALIDATIONTABLE
version	--version	-v		AS_AMDIN_VERSION
virtualserver	--virtualserver			AS_ADMIN_VIRTUALSERVER

# Failover Scenarios (Enterprise Edition)

The scenarios in this appendix illustrate how session persistence settings for application server instances and web applications affect the failover of session state and stateful session bean (SFSB) state in Sun Java System <Product Name>, Enterprise Edition. It also includes a discussion and sample failover scenario for Java Remote Method Invocation (RMI) and Internet Inter-Orb Protocol (IIOP) requests.

This appendix includes the following topics:

- [HTTP Session Failover](#)
- [Stateful Session Bean Failover](#)
- [Failover on RMI/IIOP Path](#)

## HTTP Session Failover

HTTP sessions can be failed over between server instances in a cluster. For information about setting up a cluster for failover, see [Chapter 19, “Managing Clusters \(Enterprise Edition\).”](#) For information about configuring session persistence, see [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\).”](#)

This section contains the following topics:

- [Assumptions and Requirements](#)
- [Persistence Type Set to memory for Application Server Instances in Cluster](#)
- [Persistence Type Set to memory for an Application Server Instance](#)

- [Persistence Type Set to file for an Application Server Instance](#)
- [Persistence Type Set to ha for Application Server Instances in a Cluster](#)
- [Persistence Type Set to memory for a Web Application](#)
- [Persistence Type Set to file for a Web Application](#)
- [Persistence Type Set to ha for a Web Application](#)

## Assumptions and Requirements

The following information about HTTP session persistence is relevant to the scenarios described in this appendix:

- For production environments that require HTTP session persistence, the only persistence type supported is `ha`.
- To support session persistence for HTTP sessions you must enable high availability. For more information about enabling high-availability, see [“Enabling and Disabling Availability” on page 474](#). In order for availability to work at the application level, it must have been enabled for the application and at all levels above the application. In the scenarios described here, it is assumed that availability has been enabled for the application server instances, the web containers, and the applications.
- For a web application to take part in session persistence, it should be distributable. For more information about distributable web applications, see [“Making a Web Application Distributable” on page 481](#). In the scenarios described in this appendix, it is assumed that the web applications are distributable.
- Session persistence configuration for a web application takes precedence over the session persistence configuration for the application server instances on which the web application is deployed.

For more information on session persistence and the configuration settings related to session persistence, see [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\).”](#)



## Persistence Type Set to memory for Application Server Instances in Cluster

### *Summary*

The persistence type for the application server instances in a cluster is set to `memory` and there are no persistence type settings defined for the web applications deployed in the cluster. If an application server instance on which HTTP sessions have been created becomes unavailable, the HTTP sessions created on it are not failed over to other application server instances.

### *Example*

Here is an example of the scenario:

- You have set persistence type to `memory` for all application server instances in the cluster.
- You have deployed a web application with no persistence type settings.
- This web application has serviced requests and sessions for these requests have been created on one or more application server instances.
- An application server instance on which one or more of these sessions were created becomes unavailable.

### *Result Description*

These sessions will not be failed over to other application server instances in the cluster. If requests corresponding to these sessions come in, these requests will be routed to other (available) application server instances in the cluster as per the load balancing algorithm of the load balancer.

## Persistence Type Set to memory for an Application Server Instance

### *Summary*

In a development environment, the persistence type for an application server instance is set to `memory` and there are no persistence type settings defined for the web applications deployed on the instance.

The instance is gracefully shut down. When the instance restarts, it can recover HTTP session state from a file.

This mode is useful when you are using only one application server instance and there is no load balancer. However, in a clustered environment with a load balancer, by the time you have shut down the instance completely and then restarted it, the load balancer may have failed over the session state to other instances in the cluster and the instance may not be able to recover session state when it restarts.

### *Example*

Here is an example of the scenario:

- In a development environment, you have set persistence type to `memory` for an application server instance.
- You have deployed a web application on the application server instance with no session persistence settings.
- This web application has serviced requests and sessions for these requests have been created on the instance.
- The instance is gracefully shutdown.

### *Result Description*

Before the graceful shutdown, the instance stores the session state in a file in a directory. The path of the directory is the value of the `sessionFilename` property of `session-manager` in the `server.xml` file.

When you restart the instance, it will be able to get the session state from the file.

## Persistence Type Set to file for an Application Server Instance

### *Summary*

In a development environment, the persistence type for an application server instance is set to `file` and there are no persistence type settings defined for the web applications deployed on the instance.

The instance is gracefully shutdown or becomes unavailable. When the instance restarts, it can recover HTTP session state from a file.

This mode is useful when you are using only one application server instance and there is no load balancer. However, in a clustered environment with a load balancer, the session state cannot be recovered by any other instance in the cluster. Additionally, while the instance is unavailable, the load balancer may attempt to fail sessions over to other instances, and once the instance restarts, those sessions will have to be re-established (usually requiring the web client to reauthenticate).

### *Example*

Here is an example of the scenario:

- In a development environment, you have set persistence type to `file` for an application server instance.
- You have deployed a web application on the application server instance with no session persistence settings.
- This web application has serviced requests and sessions for these requests have been created on the instance.
- The instance is gracefully shutdown or becomes unavailable.

### *Result Description*

While the application server instance is available, it stores the information for each session in a corresponding file. The session is stored at intervals specified by the `reapIntervalSeconds` property of the `session manager` element in the `server.xml` file.

If the instance restarts, it will be able to get the information for the sessions from the corresponding files. Because the session state is stored periodically (and not, for example, after every web method), there is a risk of some loss of session data, depending on when the session was last stored.

## Persistence Type Set to `ha` for Application Server Instances in a Cluster

### *Summary*

The persistence type for the application server instances in a cluster is set to `ha` and there are no persistence type settings defined for the web applications deployed in the cluster. If an application server instance on which sessions have been created becomes unavailable, the sessions created on it are failed over to other (available) application server instances in the cluster.

*Example*

Here is an example of the scenario:

- You have set persistence type to `ha` for all application server instances in the cluster.
- You have deployed a web application on the cluster with no session persistence settings.
- This web application has serviced requests and sessions for these requests have been created on one or more application server instances.
- An application server instance on which one or more of these sessions were created becomes unavailable.

*Result Description*

- The load balancer keeps monitoring all application server instances to check if they are available. When the instance becomes unavailable, the load balancer marks the instance as unavailable.
- The load balancer stops sending any unassigned requests to the instance.
- Requests that were sent to the application server instance but that were not serviced are lost and an error is sent to the client.
- All subsequent requests for sessions assigned to the application server instance are failed over to other instances in the cluster.

The instance that was originally serving the requests would have written the HTTP session state to the high-availability database (HADB).

The persistence frequency setting has a bearing on how up-to-date the HTTP session state information in the HADB is.

- If the persistence frequency for this application server instance is set to `time-based`, the HTTP session state is stored periodically at the time interval specified 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). For more information, see [“Configuring Other HTTP Session Persistence Properties” on page 502](#).
- If the persistence frequency for the application server instance is set to `web-method`, the session state is stored after every web request.

## Persistence Type Set to memory for a Web Application

### *Summary*

The persistence type for a web application deployed on a cluster is set to `memory`. If an application server instance on which sessions for this web application have been created becomes unavailable, the sessions created on it are not failed over to other application server instances in the cluster. This will happen regardless of the persistence type setting for the application server instances in the cluster.

### *Example*

Here is an example of the scenario:

- You have deployed a web application with session persistence set to `memory`.
- This web application has serviced requests and sessions for these requests have been created on one or more application server instances.
- An application server instance on which one or more of these sessions were created becomes unavailable.

### *Result Description*

The result is the same as in the case of [“Persistence Type Set to memory for Application Server Instances in Cluster”](#) on page 617.

## Persistence Type Set to file for a Web Application

### *Summary*

The persistence type for a web application deployed on a cluster is set to `file`. If an application server instance on which sessions for this web application have been created becomes unavailable, the sessions created on it are not failed over to other application server instances in the cluster. This will happen regardless of the persistence type setting for the application server instances in the cluster.

### *Example*

Here is an example of the scenario:

- You have deployed a web application with session persistence set to `file`.

- This web application has serviced requests and sessions for these requests have been created on one or more application server instances.
- An application server instance on which one or more of these sessions were created becomes unavailable.

#### *Result Description*

The result is the same as in the case of [“Persistence Type Set to file for an Application Server Instance” on page 618.](#)

## Persistence Type Set to ha for a Web Application

#### *Summary*

The persistence type for a web application deployed on a cluster is set to `ha`. If an application server instance on which sessions for this web application have been created becomes unavailable, the sessions created on it are failed over to other application server instances in the cluster. This will happen regardless of the persistence type setting for the application server instances in the cluster.

#### *Example*

Here is an example of the scenario:

- You have deployed a web application with session persistence set to `ha`.
- This web application has serviced requests and sessions for these requests have been created on one or more application server instances.
- An application server instance on which one or more of these sessions were created becomes unavailable.

#### *Result Description*

The result is the same as in the case of [“Persistence Type Set to ha for Application Server Instances in a Cluster” on page 619.](#)

# Stateful Session Bean Failover

The state of Stateful Session Beans (SFSBs) also fails over between server instances in a cluster. If the request for an SFSB comes in over the HTTP path, the stateful session bean state fails over between server instances in the HTTP cluster. If the request for an SFSB comes in over the RMI/IIOP path, the SFSB state fails over between the instances in an IIOP cluster.

This section contains the following topics:

- [Assumptions and Requirements](#)
- [SFSB State Stored in the File System](#)
- [SFSB State Stored in the HADB](#)

## Assumptions and Requirements

The following information about SFSB failover is relevant to the scenarios described in this appendix:

- To support failover of the SFSB state, you must enable high availability. High availability must be enabled for the server instance, EJB container, module, and for any SFSB for which the state fails over. For more information on enabling availability for the instance, container, and module levels, see [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\)”](#). For information on enabling availability and state checkpointing for SFSBs, see the *Sun Java System <Product Name> Developer's Guide to Enterprise JavaBeans Technology*.
- If you've specified a JNDI name of the JDBC resource for the HADB, the SFSB persistent store is assumed to be the HADB. You specify the JNDI name in the `store-pool-jndi-name` property in the `availability-service` element of the `server.xml` file. If you have not specified a JNDI name of the JDBC resource for the HADB, the persistent store defaults to the file system.
- The same persistence store (HADB or file) should be used for both the web container and the EJB container. In production environments that require failover of SFSB state, the persistence store must be the HADB and the persistence type for the server instance must be set to `ha`.

- Transparent failover of in-flight requests is not supported. If a server instance to which a client request is sent crashes during the request processing (before the response could be prepared and sent back to the client) an exception or error is sent to the client. The client must retry the request explicitly. When the client retries the request, the request is sent to another server instance in the cluster, which receives session state information for this client.

For more information on session persistence and the configuration settings related to session persistence, see [Chapter 20, “Configuring Session Persistence \(Enterprise Edition\).”](#)

## SFSB State Stored in the File System

### *Summary*

In a development environment, the persistence type for an application server instance is set to `file` or `memory` and the HADB is not configured.

The instance is gracefully shutdown or becomes unavailable. When the instance restarts, it can recover SFSB state from the file system.

This mode is useful when you are using only one application server instance and there is no load balancer. However, in a clustered environment with a load balancer, the SFSB state cannot be recovered by any other instance in the cluster.

### *Example*

Here is an example of the scenario:

- In a development environment, you have set persistence type to `file` or `memory` for an application server instance.
- The `store-pool-jndi-name` property in the `availability-service` element of the `server.xml` file is not present or its value is an empty string. This configuration sets the EJB container's persistence store to the file system.
- You have enabled availability for the server instance, EJB container, module, and SFSBs for which you want the SFSB state saved.
- A client has requested a SFSB and received a response.
- An application server instance on which one or more of these requests for SFSBs were served becomes unavailable.



*Result Description*

While the application server instance is available, it stores information on SFSB state in the file system, at the location specified in the `session-store` attribute in the `server.xml` file. The SFSB state is stored at checkpoints, either at the end of a transaction in which the SFSB participates, or at the end of a business method invoked on the SFSB.

If the instance restarts, it will be able to get the information for the SFSB states from the file system.

## SFSB State Stored in the HADB

*Summary*

The persistence type for the application server instances in a cluster is set to `ha` and the HADB is configured. Availability is enabled. If an application server instance on which sessions have been created becomes unavailable, the SFSB states on it are failed over to other (available) application server instances in the cluster.

*Example*

Here is an example of the scenario:

- You have set persistence type to `ha` for all application server instances in the cluster.
- The `store-pool-jndi-name` property in the `availability-service` element of the `server.xml` file has been configured with the JNDI name of the JDBC resource for the HADB. This configuration sets the persistent store for SFSBs to the HADB.
- You have enabled availability for the server instances, EJB containers, modules, and SFSBs for which you want the SFSB state to fail over.
- A client has requested a SFSB and received a response.
- An application server instance on which one or more of these requests for SFSBs were served becomes unavailable.

*Result Description*

- The load balancer keeps monitoring all application server instances in the cluster to check if they are available. When the instance becomes unavailable, the load balancer marks the instance as unavailable.
- The load balancer stops sending any unassigned requests to the instance.

- Requests are serviced for the amount of time specified in `disable-timeout-in-minutes` in the `loadbalancer.xml` file.
- Requests that were sent to the application server instance but that were not serviced are lost and an error is sent to the client.
- All subsequent requests for sessions assigned to the application server instance are failed over to other instances in the cluster.

The instance that was originally serving the requests would have written the SFSB state to the high-availability database (HADB). The SFSB state is stored at checkpoints, either at the end of a transaction in which the SFSB participates, or at the end of a business method invoked on the SFSB.

The application server instances to which the sessions are failed over access the SFSB state from the HADB.

## Failover on RMI/IIOP Path

RMI/IIOP request failover provides high availability of RMI/IIOP references. These references can be either EJBs or `NameService` references. If the server instance that published an RMI/IIOP remote reference becomes unavailable, additional requests for the reference are failed over to an accessible instance in the cluster. In order for RMI/IIOP requests to fail over to another server instance, RMI/IIOP endpoints must be configured for the other server instances in the cluster. For more information on clusters, see [Chapter 19, “Managing Clusters \(Enterprise Edition\).”](#)

Once an IIOP request for an object is failed over to new server instance, all the subsequent requests to that IIOP object will continue to go to the same server instance even if the original server instance is back to life. Thus, the stickiness of objects to server instances is maintained even after failover.

This section contains the following topics:

- [Assumptions and Requirements](#)
- [RMI/IIOP Failover Scenario](#)

## Assumptions and Requirements

The following information about RMI/IIOP failover is relevant to the scenario described in this appendix:

- To support RMI/IIOP failover, you must enable high availability for the server instance.
- You must have defined a cluster and configured the IIOP endpoints for the server instances in that cluster. These configuration settings are stored in the `server.xml` file. For more information, see [“Configuring HTTP Load Balancing and Failover.”](#)
- You must have configured the RMI/IIOP client. The configuration required for the client varies depending upon the type of the client. For information on configuring clients, see the *Sun Java System Application Server Developer’s Guide to Clients*.
- The following types of clients are supported for RMI/IIOP failover:
  - Java applications executing in the ACC and accessing EJBs deployed on an instance
  - Standalone clients
- For failover of SFSB state to work over IIOP, the configuration must meet the requirements outlined in [“Stateful Session Bean Failover”](#) on page 623.
- If the server instance that published a remote reference becomes inaccessible before an RMI/IIOP request is dispatched to it, the request is failed over to another server instance. If a server instance becomes unavailable while processing an RMI/IIOP request, that request will not be retried or failed over to another Application Server instance.

## RMI/IIOP Failover Scenario

### Summary

A client requests a remote reference from the server instance in a cluster which has IIOP endpoints configured. The load balancer sends the request to a particular server instance. The instance is gracefully shut down or becomes unavailable.

### *Example*

Here is an example of the scenario:

- A client application requests a remote EJB with IIOP.
- An application server instance in a cluster services the request.
- The application server instance which serviced the request becomes unavailable.
- The client makes another request for the remote reference to the unavailable server instance.

### *Result Description*

The request for the remote EJB is failed over to another server instance in the IIOP cluster.

If the request is for an SFSB, the state stored in the HADB fails over as well.

# Compiling and Configuring Apache Web Server (Enterprise Edition)

This appendix describes how you should compile the Apache source code and configure your installation of Apache Web Server to use Sun Java™ System Application Server 7, Enterprise Edition load balancer plug-in.

You must download the appropriate Apache source code for the platform on which you want to run Apache Web Server. The load balancer plug-in is certified to work with Apache on the following platforms:

- Solaris 8 & 9 SPARC
- Solaris 9 x86
- Red Hat Enterprise Linux Advanced Server 2.1

This appendix contains the following topics:

- [Minimum Requirements](#)
- [Installing SSL-aware Apache](#)

## Minimum Requirements

You must meet the following requirements to successfully compile Apache web server to run the load balancer plug-in. The Apache source must be compiled and built to run with SSL.

- openssl-0.9.7c (source)
- mod\_ssl-2.8.16-1.3.29 (source)
- apache\_1.3.29 (sources)

- `gcc-3.3-sol9-sparc-local` packages (for Solaris 9 SPARC/x86).
- `flex-2.5.4a-sol9-sparc-local` packages (for Solaris 9 SPARC)
- `flex-2.5.4a-sol9-intel-local` packages (for Solaris 9 x86)
- Sun Java System Application Server 7, Enterprise Edition installables.

Before compiling Apache on Linux, ensure that you have an installation of Sun Java System Application Server Enterprise Edition on the same machine.

- On Solaris 8, ensure that `gcc` and `make` are in the `PATH`.
- On Solaris 9, ensure that `gcc` version 3.3 and `make` are in the `PATH`, and `flex` is installed.
- If you are using `gcc` On Red Hat Enterprise Linux Advanced Server 2.1, make sure you have a version later than `gcc 3.0`.

---

**NOTE**

- To use another C compiler, set the path of the C compiler and `make` utility in the `PATH` environment variable. For example:

```
exportLD_LIBRARY_PATH=$LD_LIBRARY_PATH:appserver_installdir/lib
```

- These software sources are available at <http://www.sunfreeware.com>
- 

## Installing SSL-aware Apache

Follow these steps to compile, configure, and install SSL-aware Apache web server.

- [Compiling and Building Open SSL](#)
- [Configuring Apache with mod SSL](#)
- [Compiling and Building Apache](#)
- [Starting and Stopping Apache](#)

## Compiling and Building Open SSL

Unpack the `openssl-0.9.7c` source and follow these steps.

1. `cd openssl-0.9.7c`
2. `./config`
3. `make`

## Configuring Apache with mod SSL

Unpack the `mod_ssl-2.8.14-1.3.29` source and follow these steps.

1. `cd mod_ssl-2.8.14-1.3.29`
2. run `./configure --with-apache=../apache_1.3.27--with-ssl=../openssl-0.9.7c --prefix=<install path> --enable-module=ssl --enable-shared=ssl --enable-rule=SHARED_CORE --enable-module=so`

The directory specified in the above command is a variable. You can specify the path where you install Apache. The *prefix* argument indicates where you wish to install Apache. This command will output several lines on the screen. Essentially this command creates the `make` files for the build according to your system configuration. If there are errors in `configure`, you may be missing some header files or utility programs which you must install before proceeding.

## Compiling and Building Apache

1. Download the `apache_1.3.29` source distribution.

Unpack the source distribution. The source distribution comes as a compressed archive. If you are installing `apache_1.3.29`, the source distribution archive will read `apache_1.3.29.tar.gz`.

2. Decompress and untar the archive using the following command:

```
tar -zxvf apache_1.3.29.tar.gz
```

This command will create a directory called `apache_1.3.29` in your current working directory.

### 3. Compile Apache using the `make` command, as follows:

- a. `cd apache_1.3.29`
- b. `make`
- c. `make certificate`
- d. `make install`

This installs apache in the location provided in the `--prefix` attribute in [“Configuring Apache with mod SSL” on page 631](#).

---

#### NOTE

- On Linux, include the following lines in `src/MakeFile` after End of automatically generated section

```
LIBS+= -licuuc -licui18n -lnspr4 -lpthread -lxerces-c -lsupport
-lnsprwrap -lns-httpd40
```

```
LDLFLAGS+= -L/appserver_installdir/lib
```

- These settings must be performed before executing [Step 3 on page 632](#).
  - `make certificate` would ask for a secure password. Remember this password as it's required for starting secure Apache.
- 

The command `make install` will output several lines on the screen indicating that the process is compiling Apache source code and linking Apache. This process should normally conclude without errors. However if any errors occur, please check if all the library files and utility programs of Apache have been properly downloaded.

## Starting and Stopping Apache

Apache comes bundled with a script titled `apachectl` that facilitates starting, stopping and restarting Apache.

Run the follow command to start Apache:

```
apache_install_dir/bin/apachectl start
```

Run the follow command to start Apache in SSL mode:

```
apache_install_dir/bin/apachectl startssl
```

To stop apache, run the following command:

```
apache_install_dir/bin/apachectl stop
```



After you start, you can test your installation of Apache. Once Apache is running, type the following address in your web browser: `http://localhost/`. If your installation was successful and Apache is running, you should see a test page displaying a message to that effect.



# Running Message Queue Broker with Trial Enterprise License

Sun Java™ System Application Server 7 Standard and Enterprise Edition uses Sun Java System Message Queue Platform Edition 3.5 SP1 as the JMS provider.

This appendix describes how to run the Enterprise Edition of Sun Java System Message Queue 3.5 SP1 using a trial license. The trial license has a 90-day evaluation period.

## About Enterprise Edition

The enterprise edition of Message Queue provides the following additional features over the Platform Edition.

- Clustering and failover of JMS connections.
- Multiple Consumer Durable Subscriptions
- Support for multi-broker message services
- HTTP/HTTPS connections
- Secure connection services
- Scalable connection capability
- Multiple queue delivery policies
- Develop, debug, and load test messaging applications and components.

For more information on these and other features, see the product documentation available at: [docs.sun.com](http://docs.sun.com).

## Running Message Queue Broker with Trial License of Enterprise Edition

To run the Message Queue broker with a trial license of the Enterprise Edition, start the broker that is used by the Application Server instance with the following additional option:

```
-license try
```

---

<b>NOTE</b>	By default, when an Application Server instance is started, a Message Queue broker instance is also started.
-------------	--

---

### *Example: Using the -license try Option*

The following example shows the use of `asadmin` command to run the Message Queue broker with the trial license of Sun Java System Message Queue Enterprise Edition 3.5.

The Application Server instance in this example is named `server1`.

```
asadmin set "server1.jms-service.startArgs= -license try" --user admin  
--password mypassword --port 4848 --host localhost
```

For the change to take effect, you must restart the Application Server instance.

---

<b>NOTE</b>	Enclose the string <code>server1.jms-service.startArgs= -license try</code> in double-quotes, as shown in the example. Make this change when the Application Server instance is not running.
-------------	--

---

For more information on JMS service attributes, see [Appendix A, “Using the Command Line Interface.”](#)

## Purchasing the Enterprise Edition License

To purchase the Sun Java System Message Queue Enterprise Edition license after the evaluation period, go to:

<http://store.sun.com/>

# HTTP Load Balancer Configuration File

The `loadbalancer.xml` file contains the configuration information for load balancing Sun Java™ System Application Server 7 Enterprise Edition instances. This file is part of the load balancer plug-in for the front-end web server. Although located in the web server, it is described in this manual because of its importance and relevance to the Sun Java System Application Server.

The encoding of the `loadbalancer.xml` file is UTF-8 to maintain compatibility with regular UNIX text editors. A schema file, `sun-loadbalancer_1_1.dtd`, determines the format and content of the `loadbalancer.xml` file.

This chapter describes `loadbalancer.xml` and `sun-loadbalancer_1_1.dtd` files in these sections:

- [sun-loadbalancer\\_1\\_1.dtd File](#)
- [Elements in the loadbalancer.xml File](#)

## sun-loadbalancer\_1\_1.dtd File

The `sun-loadbalancer_1_1.dtd` file defines the structure of the `loadbalancer.xml` file, including the elements it can contain and the subelements and attributes these elements can have.

---

<b>NOTE</b>	Do not edit the <code>sun-loadbalancer_1_1.dtd</code> file; its contents change only with new versions of Sun Java System Application Server Enterprise Edition.
-------------	--

---

---

**NOTE** The `sun-loadbalancer_1_1.dtd` interface is unstable. An unstable interface may be experimental or transitional, and hence may change incompatibly, be removed, or be replaced by a more stable interface in the next release.

---

For general information about DTD files and XML, see the XML specification at:

<http://www.w3.org/TR/REC-xml>

Each element defined in a DTD file (which may be present in the corresponding XML file) can contain the following:

- [Subelements](#)
- [Data](#)
- [Attributes](#)

## Subelements

Elements can contain subelements. For example, the following file fragment defines the `cluster` element.

```
<!ELEMENT cluster (instance*, web-module*, health-checker?)>
```

The `ELEMENT` tag specifies that a `cluster` element can contain `instance`, `web-module`, and `health-checker` elements in that order.

The following table shows how optional suffix characters of subelements determine the requirement rules, or number of allowed occurrences, for the subelements. The left column lists the subelement ending character, and the right column lists the corresponding requirement rule.

**Table E-1** requirement rules and subelement suffixes

Subelement Suffix	Requirement Rule
<i>element*</i>	Can contain <i>zero or more</i> of this subelement.
<i>element?</i>	Can contain <i>zero or one</i> of this subelement.
<i>element+</i>	Must contain <i>one or more</i> of this subelement.
<i>element</i> (no suffix)	Must contain <i>only one</i> of this subelement.

If an element cannot contain other elements, you see `EMPTY` or `(#PCDATA)` instead of a list of element names in parentheses.

## Data

Some elements contain character data instead of subelements. These elements have definitions of the following format:

```
<!ELEMENT element-name (#PCDATA)>
```

For example:

```
<!ELEMENT description (#PCDATA)>
```

In the `loadbalancer.xml` file, white space is treated as part of the data in a data element. Therefore, there should be no extra white space before or after the data delimited by a data element. For example:

```
<description>response timeout property</description>
```

## Attributes

Elements that have `ATTLIST` tags contain attributes (name-value pairs). For example:

```
<!ATTLIST instance    name          CDATA      #REQUIRED
                      enabled        %boolean;  "true"
                      disable-timeout-in-minutes CDATA  "31"
                      listeners      CDATA      #REQUIRED>
```

An instance element can contain `name`, `enabled`, `disable-timeout-in-minutes`, and `listeners` attributes.

The `#REQUIRED` label means that a value must be supplied. The `#IMPLIED` label means that the attribute is optional, and that Sun Java System Application Server generates a default value. Wherever possible, explicit defaults for optional attributes (such as `"true"`) are listed.

Attribute declarations specify the type of the attribute. For example, `CDATA` means character data, and `%boolean` is a predefined enumeration.

## Elements in the loadbalancer.xml File

This section describes the XML elements in the `loadbalancer.xml` file.

<b>NOTE</b>	Subelements must be defined in the order in which they are listed under each <b>Subelements</b> heading unless otherwise noted.
-------------	---

## loadbalancer

Defines a load balancer. This is the root element; there can only be one `loadbalancer` element in a `loadbalancer.xml` file.

### Subelements

The following table describes subelements for the `loadbalancer` element. The left column lists the subelement name, the middle column indicates the requirement rule, and the right column describes what the element does.

**Table E-2** `loadbalancer` subelements

Element	Required	Description
<code>cluster</code>	zero or more	Defines a cluster of server instances.
<code>property</code>	zero or more	Specifies a load balancer property.

### Attributes

none

## property

Specifies a load balancer property.

### Subelements

The following table describes subelements for the `property` element. The left column lists the subelement name, the middle column indicates the requirement rule, and the right column describes what the element does.

**Table E-3** `property` subelements

Element	Required	Description
<code>description</code>	zero or one	Contains a text description of this element.



**Attributes**

The following table describes attributes for the `property` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes what the attribute does.

**Table E-4**    `property` attributes

Attribute	Default	Description
name	none	Specifies the name of the property.
value	none	Specifies the value of the property.

**Properties**

The following table describes load balancer properties. The left column lists the property name, the middle column indicates the default value if the property is not present, and the right column describes what the property does.

**Table E-5**    Load Balancer Properties

Property Name	Default Value	Description
response-timeout-in-seconds	60	Specifies the timeout interval within which a response must be obtained for a request that is load balanced, or an error page will be sent to the client.
reload-poll-interval-in-seconds	0	Specifies the interval at which the load balancer checks whether the <code>loadbalancer.xml</code> time stamp has changed. If it has changed, the load balancer reloads the file. If 0, polling is disabled.
https-routing	false	Specifies whether the load balancer routes incoming HTTPS requests to the instance as HTTPS requests.
require-monitor-data	false	Specifies whether load balancer monitoring is enabled.

**description**

Contains a text description of the parent element.

**Subelements**

none

**Attributes**

none

# cluster

Defines a cluster of server instances.

## Subelements

The following table describes subelements for the `cluster` element. The left column lists the subelement name, the middle column indicates the requirement rule, and the right column describes what the element does.

**Table E-6**    `cluster` subelements

Element	Required	Description
<code>instance</code>	zero or more	Defines a server instance.
<code>web-module</code>	zero or more	Defines a web module.
<code>health-checker</code>	zero or one	Configures the cluster's health checker.

## Attributes

The following table describes attributes for the `cluster` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes what the attribute does.

**Table E-7**    `cluster` attributes

Attribute	Default	Description
<code>name</code>	<code>none</code>	Specifies the name of the cluster. Within a load balancer, cluster names must be unique.

# instance

Defines a server instance.

## Subelements

`none`

## Attributes

The following table describes attributes for the `instance` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes what the attribute does.

**Table E-8** instance attributes

Attribute	Default	Description
name	none	Specifies the name of the server instance. Instance names must be unique within a cluster as well as across clusters.
enabled	true	(optional) Specifies whether instance is active (enabled) for requests to be load balanced to it.
disable-timeout-in-minutes	31	(optional) Specifies the quiescing timeout interval, at which the load balancer disables the instance and closes all its open connections.
listeners	none	Specifies a space-delimited list of the URLs of the instance's listeners. Within a cluster, as well as across clusters, listener URLs must be unique.

## web-module

Defines a web module.

### Subelements

The following table describes subelements for the `web-module` element. The left column lists the subelement name, the middle column indicates the requirement rule, and the right column describes what the element does.

**Table E-9** web-module subelements

Subelement	Required	Description
idempotent-url-pattern	zero or more	Defines the idempotent URLs to be retried.

### Attributes

The following table describes attributes for the `web-module` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes what the attribute does.

**Table E-10** web-module attributes

Attribute	Default	Description
context-root	none	<p>Specifies the context root of the web module. Within a cluster, context roots must be unique.</p> <p>The default context root can be specified as “ ” or “/”. Context roots can also be specified with a leading slash (/).</p> <p>In a cluster, if an empty context root is used along with specific context roots, the load balancer plug-in will give precedence to the specific context roots.</p>
enabled	true	(optional) Specifies whether web module is active (enabled) for requests to be load balanced to it.
disable-timeout-in-minutes	31	(optional) Specifies the quiescing timeout interval, at which the load balancer disables the web module and closes all its open connections.
error-url	none	Specifies the page to be displayed in case of a failure / error. If the error-url sub-element is missing or invalid, the load balancer will display the default error page, sun-http-lb-error.html, from the errorpages directory of the load balancer plug-in.

## idempotent-url-pattern

Defines the URL pattern of requests that the load balancer can retry in case of a failure, on healthy server instances.

### Attributes

The following table describes attributes for the idempotent-url-pattern subelement. The left column lists the attribute name, the middle column indicates the default value, and the right column describes what the attribute does.

**Table E-11** idempotent-url-pattern attributes

Attribute	Default	Description
url-pattern	none	The url-pattern is either an absolute URL for the idempotent request or an URL specification that uses wild cards.
no-of-retries	-1	<p>Attribute to define the number of retries the load balancer must perform for an idempotent request. If the value is -1, the number of retries will be equal to one less than the number of Application Server instances that can serve this request.</p> <p>The Application Server instances that can server this request can span multiple clusters, provided all the instances have the same context root.</p>

# health-checker

Configures the cluster's health checker.

## Subelements

none

## Attributes

The following table describes attributes for the `health-checker` element. The left column lists the attribute name, the middle column indicates the default value, and the right column describes what the attribute does.

**Table E-12** Health checker properties

Attribute	Default	Description
<code>url</code>	<code>/</code>	(optional) Specifies the URL to ping to determine the health state of a listener.
<code>interval-in-seconds</code>	<code>30</code>	(optional) Specifies the interval at which health checks of instances occur.
<code>timeout-in-seconds</code>	<code>10</code>	(optional) Specifies the timeout interval within which a response must be obtained for a listener to be considered healthy.



# Using the cladmin Command for Administration (Enterprise Edition)

This chapter describes the use of the `cladmin` command to administer Sun Java™ System Application Server clusters.

This module contains the following sections:

- [About the cladmin Command](#)
- [Cluster Operations Using cladmin Command](#)
- [asadmin Commands Supported by cladmin](#)
- [Requirements for Running the cladmin Command](#)
- [Using cladmin Command](#)
- [Limitations of cladmin Command](#)

## About the cladmin Command

You can use the `cladmin` command to run `asadmin` commands together on all Application Server instances in a cluster. For example, you can use this command to start all instances in a cluster or to deploy a web application to all instances in a cluster.

The `cladmin` command needs input files for its functioning. The instance-specific information is stored in the `cinstance.conf` file and the information about password is stored in the `clpassword.conf` file. You can customize the input files entries to support different configurations. Both the input files are text files. More information about the input files is provided in [“Using cladmin Command” on page 650](#).

The command reads the instance-specific information from the `cinstance.conf` file one instance at a time. You supply the rest of the argument to the `asadmin` command. The command will invoke the corresponding `asadmin` command on each instance specified in the `cinstance.conf` file.

The command is located in `InstallDir/bin` where `InstallDir` is the location of the directory where Sun Java System Application Server is installed. The input files are located in the configuration directory of the Sun Java System Application Server. The default location of the configuration directory is `etc/opt/SUNWappserver7`.

## Cluster Operations Using cladmin Command

You can perform the following operations on Application Server instances in a cluster using `cladmin`:

- Start all instances in a cluster
- Stop all instances in a cluster
- Deploy an application to all Application Server instances in a cluster
- Undeploy an application from all Application Server instances in a cluster
- Create a JDBC resource for all Application Server instances in a cluster
- Create a JDBC connection pool for all Application Server instances in a cluster
- Configure session persistence for all Application Server instances in a cluster
- Configure session persistence for all Application Server instances in a cluster
- Delete the JDBC connection pool for all Application Server instances in a cluster
- Delete the JDBC resource for all Application Server instances in a cluster

## asadmin Commands Supported by cladmin

The following table lists `asadmin` commands that you can run simultaneously for each application server instance in a cluster using the `cladmin` command. In the table, the left column lists the `asadmin` command and the right column lists the purpose of the command.



**Table F-1** asadmin Commands Supported for Use with cladmin Command

Command	Use
start-instance	Starts the application server instance.
stop-instance	Stops the application server instance.
deploy	Deploys an EJB, WEB, connector, appclient, or application component that is in the directory to the application server instance.
undeploy	Removes the deployed component from the application server instance.
create-jdbc-resource	Creates a JDBC resource for the application server instance.
create-jdbc-connection-pool	Creates a JDBC connection pool for the application server instance.
configure-session-persistence	Configures session persistence for the application server instance.
delete-jdbc-resource	Deletes the JDBC resource for the application server instance.
delete-jdbc-connection-pool	Deletes the JDBC connection pool for the application server instance.

## Requirements for Running the cladmin Command

You must meet the following requirements before using cladmin command:

- All instances that are part of the cluster must have the same user name and password for the administrator.
- Before running the command you must start the administration servers of all application server instances that are part of the cluster.
- The values specified in the input files should be same for all the instances in a cluster. The `cladmin` command is not designed to setup each instance with different values. For example, this command cannot create a JDBC Connection Pool with different settings for each instance.
- You must have performed the HADB shared memory set up and the HADB Cluster Host Communication Set up for SSH and RSH. For more information, see chapter, Post-installation Tasks, (or whatever it will be) in Sun Java System Application Server Installation Guide.

# Using cladmin Command

This section describes the following topics:

- [Input Files for the cladmin Command](#)
- [Running the cladmin Command](#)
- [cladmin Status Messages](#)
- [Log file for the cladmin Command](#)

## Input Files for the cladmin Command

The input files that are needed for the `cladmin` command to function are:

- [clinstance.conf File](#)

This file contains information about the Application Server instances that are part of the cluster

- [clpassword.conf File](#)

This file contains information about the asadmin password.

### clinstance.conf File

For the `cladmin` command to work properly, all application server instances that are part of a cluster must be defined in the `clinstance.conf` file

During standard installation, the installation program creates a `clinstance.conf` file with entries for two instances. The default values for these entries are provided in [“Entries in the clinstance.conf File” on page 651](#).

If you add more instances to the cluster, you must add information about these instances in the `clinstance.conf` file.

If you create more clusters, you must specify the information about the instances in each cluster in a separate `clinstance.conf` file.

The format of the `clinstance.conf` file is described here. One set of entries is required for each instance that is part of the cluster. Any entry that has # as the first character is treated as a comment.

---

**CAUTION** The order in which these entries appear in the `clinstance.conf` file is important and must not be changed from the order shown here. If you add information about more application server instances, entries for these instances must appear in the order shown here

However, you can add comments anywhere in the file.

---

```
# Comment
instancetype Instance Name
user User Name
host localhost
port Admin Port Number
domain Domain Name
instanceport Instance Port Number
```

The following table provides information about each possible entry in the `clinstance.conf` file, its meaning, and its default value. The default value is the value specified by the installation program during a standard installation for the two entries that it creates.

In the table, the left column lists the entry in the `clinstance.conf` file, the middle column provides the definition of the entry, and the right column lists the default value(s) of the entry.

**Table F-2** Entries in the `clinstance.conf` File

Entry	Definition	Default Value(s)
Instance Name	Name of the application server instance	server1, server2
User Name	Administration server user name	admin
Localhost	Host name	localhost
Admin Port Number	Port number of the Administration Server	4848
Domain Name	Name of the Administrative domain	domain1

**Table F-2** Entries in the clinstance.conf File (*Continued*)

Entry	Definition	Default Value(s)
Instance Port Number	Port number of the application server instance	80, 81

Here is a sample `clinstance.conf` file with information about two instances.

```
#Instance 1
instancename server1
user admin
host localhost
port 4848
domain domain1
instanceport 80

#Instance 2
instancename server2
user admin
host localhost
port 4848
domain domain1
instanceport 81
```

**clpassword.conf File**

The `clpassword.conf` file contains the administration server password. During the execution of the `cladmin` command, the `asadmin` command requires the administration server password. The password specified in the `clpassword.conf` file is used.

The format of the `clpassword.conf` file is as follows

```
AS_ADMIN_PASSWORD= password
```

where *password* is the `asadmin` password.

If you ran the `clsetup` command as part of the post installation step, the `clpassword.conf` file would have been created and should be available in the configuration directory of the Sun Java System Application Server. The value of *password* will also be entered in the file. Permissions 0600 are preset on this file and the file can be accessed only by the root user.

If you did not run the `clsetup` command as part of the post installation step, you must create this file manually and save it in the configuration directory of the Sun Java System Application Server. You must also enter the value for password and set appropriate permissions on the file.

## Running the cladmin Command

Before running the `cladmin` command:

- Make sure you list all the instances that are part of the cluster in the `clinstance.conf` file.
- Change the directory to `install_dir/bin` where *install\_dir* is the Sun Java System Application Server installation directory.

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.

---

**NOTE** The input files `clinstance.conf` and `clpassword.conf` must follow the required format as specified in [“Using cladmin Command” on page 650](#).

---

If the input files are located in the Configuration directory of the Sun Java System Application Server (by default `/etc/opt/SUNWappserver7`), you can omit the `instancefile` and `passwordfile` options and run the command as follows:

```
./cladmin asadmin_command
```

In this case, the input files will be read from the configuration directory of the Sun Java System Application Server.

**Code Example F-1**      Example: Starting All Instances in a Cluster

```
./cladmin start-instance
```

Because the location of the input files has not been specified, the input files are read from the Configuration directory of the Sun Java System Application Server (by default /etc/opt/SUNWappserver7)

**Code Example F-2**      Example: Starting All Instances in a Cluster with Location of Input Files *Specified*

```
./cladmin --instancefile /tmp/clinstance.conf --passwordfile  
/tmp/clpassword.conf start-instance
```

**Code Example F-3**      Example: Creating a JDBC Connection Pool Named CluJDBC for *All Instances in a Cluster*

```
./cladmin create-jdbc-connection-pool --user admin  
--datasourceclassname com.sun.hadb.jdbc.ds.HadbDataSource  
--isolationguaranteed=true --isolationlevel repeatable-read  
--isconnectvalidatereq=true --validationmethod auto-commit  
--failconnection=false --property  
username=hadbuser:password=hadbpassword:serverList=exampleserver  
1.example.com\:15100,exampleserver2.example.com\:15120 CluJDBC
```

**Code Example F-4**      Example: Configuring Session Persistence for all Instances in a Cluster

```
./cladmin configure-session-persistence --user admin --type ha  
--frequency web-method --scope session --store jdbc_hastore
```

---

**NOTE**      If you are running multiple clusters, you must specify a separate `clinstance.conf` file for each cluster.

---

## cladmin Status Messages

The command returns exit codes that will help you in getting information about the command execution. The exit codes can be obtained by running

```
echo $?
```

from the command line immediately after running the `cladmin` command.

The exit codes are described in the following table. In the table, the left column lists the exit code and the right column lists the description of the exit code.

**Table F-3** Exit Codes for the `cladmin` Command

Exit Code	Description
1	Successful exit
2	Syntax error
3	Cannot find instance input file ( <code>clinstance.conf</code> )
4	Cannot read instance input file ( <code>clinstance.conf</code> )
5	Cannot find password input file ( <code>clpassword.conf</code> )
6	Cannot read password input file ( <code>clpassword.conf</code> )
7	User trying to run the command is not a root user
8	Cannot locate <code>asadmin</code> command
9	Cannot create temporary file
10	Command could not be executed

## Log file for the `cladmin` Command

The log file is named `cladmin.log` and is available in the directory, `/var/tmp/cladmin.log`

At the end of every execution, you are advised about the location of the log file. It is recommended that you scan the log file after each run.

By default the `cladmin` command executes in the verbose mode and logs all the information in the log file. Log file entries start and end with timestamp tags. If the log file exists prior to execution of the command, the output is appended to the existing log file.

## Limitations of `cladmin` Command

The following list describes the limitations of the `cladmin` command:

- It does not perform any RSH or SSH configuration.
- It does not perform any `inetd` configuration.

- It does not set any shared memory values
- Cleanup or undo scripts for the cladmin command are not provided.
- If the entities to be handled (database nodes, connection pool, and so on) exist on the machine, the cladmin command does not delete or reconfigure them. The information is redirected to the log file and the respective configuration step are skipped.
- The default input files are deleted as part of standard uninstallation. Therefore, if you want to retain the input files after uninstallation, you must back up these files before uninstallation.
- The command does not administer the High Availability Database (HADB) or the loadbalancer plug-in. For more information, see [Chapter 21, “Administering the High-Availability Database \(Enterprise Edition\).”](#)



# Glossary

This glossary provides definitions for common terms used to describe the Sun Java System Application Server deployment and development environment. For a glossary of standard J2EE terms, please see the J2EE glossary at:

<http://java.sun.com/j2ee/glossary.html>

**access control** The means of securing your Sun Java System Application Server by controlling who and what has access to it.

**ACL** Access Control List. ACLs are text files that contain lists identifying who can access the resources stored on your Sun Java System Application Server. *See also* [general ACL](#).

**activation** The process of transferring an enterprise bean's state from secondary storage to memory.

**active node** An HADB node that contains session data. If an active node fails, a spare node copies data from the mirror node and becomes active. *See also* [HADB node](#), [spare node](#), [mirror node](#), and [data redundancy unit \(DRU\)](#).

**Administration interface** The set of browser based forms used to configure and administer the Sun Java System Application Server. *See also* [CLI](#).

**administration server** An application server instance dedicated to providing the administrative functions of the Sun Java System Application Server, including deployment, browser-based administration, and access from the command-line interface (CLI) and Integrated Development Environment (IDE).

**administrative domain** Multiple administrative domains is a feature within the Sun Java System Application Server that allows different administrative users to create and manage their own domains. A domain is a set of instances, created using a common set of installed binaries in a single system.

**API** Applications Program Interface. A set of instructions that a computer program can use to communicate with other software or hardware that is designed to interpret that API.

**applet** A small application written in Java that runs in a web browser. Typically, applets are called by or embedded in web pages to provide special functionality. By contrast, a *servlet* is a small application that runs on a server.

**application** A group of components packaged into an `.ear` file with a J2EE application deployment descriptor. *See also* [component](#), [module](#).

**application disabling (in cluster)** The process by which an application deployed on a cluster is given enough time to serve the existing requests that it is handling before the load balancer stops sending requests to it. *See also* [application enabling \(in cluster\)](#), [quiescing](#).

**application enabling (in cluster)** The process of adding an application to a cluster so that the load balancer starts sending requests to the web application. *See also* [application disabling \(in cluster\)](#).

**application client container** *See* [container](#).

**application server** A reliable, secure, and scalable software platform in which business applications are run. Application servers typically provide high-level services to applications, such as component lifecycle, location, and distribution and transactional resource access.

**application tier** A conceptual division of a J2EE application. *Client tier*: the user interface (UI). End users interact with client software (such as a web browser) to use the application. *Server tier*: the business logic and presentation logic that make up your application, defined in the application's components. *Data tier*: the data access logic that enables your application to interact with a data source.

**assembly** The process of combining discrete components of an application into a single unit that can be deployed. *See also* [deployment](#).

**assigned request** A request for an existing session. Once the session is created on an application server instance in a cluster, the load balancer sends the subsequent requests corresponding to the same session only to that particular application server instance.

**asynchronous communication** A mode of communication in which the sender of a message need not wait for the sending method to return before it continues with other work.

**attribute** A name-value pair in a request object that can be set by a servlet. Also a name-value pair that modifies an element in an XML file. Contrast with *parameter*. More generally, an attribute is a unit of metadata.

**auditing** The method(s) by which significant events are recorded for subsequent examination, typically in error or security breach situations.

**authentication** The process by which an entity (such as a user) proves to another entity (such as an application) that it is acting on behalf of a specific identity (the user's security identity). Sun Java System Application Server supports basic, form-based, and SSL mutual authentication. *See also* [client authentication](#), [digest authentication](#), [host-IP authentication](#), [pluggable authentication](#).

**authorization** The process by which access to a method or resource is determined. Authorization in the J2EE platform depends upon whether the user associated with a request through authentication is in a given security role. For example, a human resources application may authorize managers to view personal employee information for all employees, but allow employees to only view their own personal information.

**availability service** The Sun Java System Application Server feature for enabling high availability on the server instance, web container, EJB container, and also for RMI/IIOP requests.

**backup store** A repository for data, typically a file system or database. A backup store can be monitored by a background thread (or sweeper thread) to remove unwanted entries.

**bean-managed persistence** Data transfer between an entity bean's variables and a data store. The data access logic is typically provided by a developer using Java Database Connectivity (JDBC) or other data access technologies. *See also* [container-managed persistence](#).

**bean-managed transaction** Where transaction demarcation for an enterprise bean is controlled programmatically by the developer. *See also* [container-managed transaction](#).

**BLOB** Binary Large Object. A data type used to store and retrieve complex object fields. BLOBs are binary or serializable objects, such as pictures, that translate into large byte arrays, which are then serialized into container-managed persistence fields.

**BMP** *See* [bean-managed persistence](#).

**BMT** *See* [bean-managed transaction](#).

**broker** The Sun Java System Message Queue entity that manages JMS message routing, delivery, persistence, security, and logging, and which provides an interface that allows an administrator to monitor and tune performance and resource use.

**business logic** The code that implements the essential business rules of an application, rather than data integration or presentation logic.

**CA** *See* [certificate authority](#), [connector architecture](#).

**cached rowset** A `CachedRowSet` object permits you to retrieve data from a data source, then detach from the data source while you examine and modify the data. A cached row set keeps track both of the original data retrieved, and any changes made to the data by your application. If the application attempts to update the original data source, the row set is reconnected to the data source, and only those rows that have changed are merged back into the database.

**Cache Control Directives** Cache-control directives are a way for Sun Java System Application Server to control what information is cached by a proxy server. Using cache-control directives, you override the default caching of the proxy to protect sensitive information from being cached, and perhaps retrieved later. For these directives to work, the proxy server must comply with HTTP 1.1.

**callable statement** A class that encapsulates a database procedure or function call for databases that support returning result sets from stored procedures.

**certificate** Digital data that specifies the name of an individual, company, or other entity, and certifies that the public key included in the certificate belongs to that entity. Both clients and servers can have certificates.

**certificate authority** A company that sells certificates over the Internet, or a department responsible for issuing certificates for a company's intranet or extranet.

**cipher** A cryptographic algorithm (a mathematical function), used for encryption or decryption.

**CKL** Compromised Key List. A list, published by a certificate authority, that indicates any certificates that either client users or server users should no longer trust. In this case, the key has been compromised. *See also* [CRL](#).

**classloader** A Java component responsible for loading Java classes according to specific rules. *See also* [classpath](#).

**classpath** A path that identifies directories and JAR files where Java classes are stored. *See also* [classloader](#).

**CLI** Command-line interface. An interface that enables you to type executable instructions at a user prompt. *See also* [Administration interface](#).

**client authentication** The process of authenticating client certificates by cryptographically verifying the certificate signature and the certificate chain leading to the CA on the trust CA list. *See also* [authentication](#), [certificate authority](#).

**client contract** A contract that determines the communication rules between a client and the EJB container, establishes a uniform development model for applications that use enterprise beans, and guarantees greater reuse of beans by standardizing the relationship with the client.

**cluster** A group of application server instances that work together as one logical entity.

**CMP** *See* [container-managed persistence](#).

**CMR** *See* [container-managed relationship](#).

**CMT** *See* [container-managed transaction](#).

**co-locate** To position a component in the same memory space as a related component in order avoid remote procedure calls and improve performance.

**column** A field in a database table.

**commit** To complete a transaction by sending the required commands to the database. *See* [rollback](#), [transaction](#).

**component** A web application, enterprise bean, message-driven bean, application client, or connector. *See also* [application](#), [module](#).

**component contract** A contract that establishes the relationship between an enterprise bean and its container.

**configuration** The process of tuning the server or providing metadata for a component. Normally, the configuration for a specific component is kept in the component's deployment descriptor file. *See also* [administration server](#), [deployment descriptor](#).

**connection factory** An object that produces connection objects that enable a J2EE component to access a resource. Used to create JMS connections (TopicConnection or QueueConnection) which allow application code to make use of the provided JMS implementation. Application code uses the JNDI Service to locate connection factory objects using a JNDI Name.

**Connection Pool** allows highly efficient access to a database by caching and reusing physical connections, thus avoiding connection overhead and allowing a small number of connections to be shared between a large number of threads. *See also* [JDBC connection pool](#)

**connector** A standard extension mechanism for containers to provide connectivity to EISs. A connector is specific to an EIS and consists of a resource adapter and application development tools for EIS connectivity. The resource adapter is plugged in to a container through its support for system level contracts defined in the connector architecture.

**connector architecture** An architecture for the integration of J2EE applications with EISs. There are two parts to this architecture: a EIS vendor-provided resource adapter and a J2EE server that allows this resource adapter to plug in. This architecture defines a set of contracts that a resource adapter has to support to plug in to a J2EE server, for example, transactions, security and resource management.

**container** An entity that provides life cycle management, security, deployment, and runtime services to a specific type of J2EE component. Sun Java System Application Server provides web and EJB containers, and supports application client containers. *See also* [component](#).

**container-managed persistence** Where the EJB container is responsible for entity bean persistence. Data transfer between an entity bean's variables and a data store, where the data access logic is provided by the Sun Java System Application Server. *See also* [bean-managed persistence](#).

**container-managed relationship** A relationship between fields in a pair of classes where operations on one side of the relationship affect the other side.

**container-managed transaction** Where transaction demarcation for an enterprise bean is specified declaratively and automatically controlled by the EJB container *See also* [bean-managed transaction](#).

**control descriptor** A set of enterprise bean configuration entries that enable you to specify optional individual property overrides for bean methods, plus enterprise bean transaction and security properties.

**conversational state** Where the state of an object changes as the result of repeated interactions with the same client. *See also* [persistent state](#).

**cookie** A small collection of information that can be transmitted to a calling web browser, then retrieved on each subsequent call from that browser so the server can recognize calls from the same client. Cookies are domain-specific and can take advantage of the same web server security features as other data interchange between your application and the server.

**CORBA** Common Object Request Broker Architecture. A standard architecture definition for object-oriented distributed computing.

**COSNaming Service** An an IIOP-based naming service.

**CosNaming provider** To support a global JNDI name space (accessible to IIOP application clients), Sun Java System Application Server includes J2EE based CosNaming provider which supports binding of CORBA references (remote EJB references).

**create method** A method for customizing an enterprise bean at creation.

**CRL** Certificate Revocation List. A list, published by a certificate authority, that indicates any certificates that either client users or server users should no longer trust. In this case, the certificate has been revoked. *See also* [CKL](#).

**data access logic** Business logic that involves interacting with a data source.

**database** A generic term for Relational Database Management System (RDBMS). A software package that enables the creation and manipulation of large amounts of related, organized data.

**database connection** A database connection is a communication link with a database or other data source. Components can create and manipulate several database connections simultaneously to access data.

**data redundancy unit (DRU)** A set of HADB nodes containing half of the active and spare nodes, and one complete copy of the data. The HADB is organized into two Data Redundancy Units (DRUs), which mirror each other. To ensure fault tolerance, the computers that support one DRU must be completely self-supported with respect to power, processing units, and storage. *See also* [HADB node](#), [active node](#), [spare node](#), and [mirror node](#).

**data source** A handle to a source of data, such as a database. Data sources are registered with the Sun Java System Application Server and then retrieved programmatically in order to establish connections and interact with the data source. A data source definition specifies how to connect to the source of data.

**DataSource Object** A DataSource object has a set of properties that identify and describe the real world data source that it represents.

**declarative security** Declaring security properties in the component's configuration file and allowing the component's container (for instance, a bean's container or a servlet engine) to manage security implicitly. This type of security requires no programmatic control. Opposite of [programmatic security](#). *See* [container-managed persistence](#).

**declarative transaction** *See* [container-managed transaction](#).

**decryption** The process of transforming encrypted information so that it is intelligible again.

**delegation** An object-oriented technique for using the composition of objects as an implementation strategy. One object, which is responsible for the result of an operation, delegates the implementation to another object, its delegatee. For example, a classloader often delegates the loading of some classes to its parent.

**deployment** The process of distributing the files required by an application to an application server to make the application available to run on the application server. *See also* [assembly](#).



**deployment descriptor** An XML file provided with each module and application that describes how they should be deployed. The deployment descriptor directs a deployment tool to deploy a module or application with specific container options and describes specific configuration requirements that a deployer must resolve.

**destination resource** An objects that represents Topic or Queue destinations. Used by applications to read/write to Queues or publish/subscribe to Topics. Application code uses the JNDI Service to locate JMS resource objects using a JNDI Name.

**digest authentication** A for of authentication that allows the user to authenticate based on user name and password without sending the user name and password as cleartext.

**digital signature** an electronic security mechanism used to authenticate both a message and the signer.

**directory server** See [Sun Java System Directory Server](#).

**Distinguished Name** See [DN](#), [DN attribute](#).

**distributable application** An application that can take part in session persistence. For an application to be highly available, it must be distributable.

**distributable session** A user session that is distributable among all servers in a cluster.

**distributed transaction** A single transaction that can apply to multiple heterogeneous databases that may reside on separate servers.

**Document Root** The document root (sometimes called the primary document directory) is the central directory that contains all the virtual server's files you want to make available to remote clients.

**Domain Registry** The Domain Registry is a single data structure that contains domain-specific information, for all the domains created and configured on an installation of Sun Java System Application Server, such as domain name, domain location, domain port, domain host.

**DN** Distinguished Name. The string representation for the name of an entry in a directory server.

**DN attribute** Distinguished Name attribute. A text string that contains identifying information for an associated user, group, or object.

**double failure** Simultaneous failure of one or more mirror node pairs in the HADB. See [High-Availability Database \(HADB\)](#), [HADB node](#), [active node](#), [spare node](#), [mirror node](#), and [data redundancy unit \(DRU\)](#).

**DTD** Document Type Definition. A description of the structure and properties of a class of XML files.

**dynamic reconfiguration** The configuration in the `loadbalancer.xml` file that, if enabled, triggers the load balancer to periodically check whether the `loadbalancer.xml` file has changed. If the load balancer detects changes to the time stamp of the `loadbalancer.xml` file, it reloads the entire configuration in the `loadbalancer.xml` file.

**dynamic redeployment** The process of redeploying a component without restarting the server.

**dynamic reloading** The process of updating and reloading a component without restarting the server. By default, servlet, JavaServer Page (JSP), and enterprise bean components can be dynamically reloaded. Also known as versioning.

**EAR file** Enterprise ARchive file. An archive file that contains a J2EE application. EAR files have the `.ear` extension. See also [JAR file](#).

**e-commerce** Electronic commerce. A term for business conducted over the Internet.

**EIS** Enterprise Information System. This can be interpreted as a packaged enterprise application, a transaction system, or a user application. Often referred to as an EIS. Examples of EISs include: R/3, PeopleSoft, Tuxedo, and CICS.

**EJB container** See [container](#).

**EJB QL** EJB Query Language. A query language that provides for navigation across a network of entity beans defined by container-managed relationships.

**EJB technology** An enterprise bean is a server-side component that encapsulates the business logic of an application. The business logic is the code that fulfills the purpose of the application. In an inventory control application, for example, the enterprise beans might implement the business logic in methods called `checkInventoryLevel` and `orderProduct`. By invoking these methods, remote clients can access the inventory services provided by the application. *See also* [container](#), [entity bean](#), [message-driven bean](#), and [session bean](#).

**ejbc utility** The compiler for enterprise beans. It checks all EJB classes and interfaces for compliance with the EJB specification, and generates stubs and skeletons.

**element** A member of a larger set; for example, a data unit within an array, or a logic element. In an XML file, it is the basic structural unit. An XML element contains subelements or data, and may contain attributes.

**encapsulate** To localize knowledge within a module. Because objects encapsulate data and implementation, the user of an object can view the object as a black box that provides services. Instance variables and methods can be added, deleted, or changed, but if the services provided by the object remain the same, code that uses the object can continue to use it without being rewritten.

**encryption** The process of transforming information so it is unintelligible to anyone but the intended recipient.

**endpoint** *See* [IIOP endpoint](#).

**entity bean** An enterprise bean that relates to physical data, such as a row in a database. Entity beans are long lived, because they are tied to persistent data. Entity beans are always transactional and multi-user aware. *See* [message-driven bean](#), [read-only bean](#), [session bean](#).

**ERP** Enterprise Resource Planning. A multi-module software system that supports enterprise resource planning. An ERP system typically includes a relational database and applications for managing purchasing, inventory, personnel, customer service, shipping, financial planning, and other important aspects of the business.

**event** A named action that triggers a response from a module or application.

**external JNDI resource** Allows the JNDI Service to act as a bridge to a remote JNDI server.

**facade** Where an application-specific stateful session bean is used to manage various Enterprise JavaBeans (EJBs).

**factory class** A class that creates persistence managers. *See also* [connection factory](#).

**failover** A recovery process where a bean can transparently survive a server crash.

**finder method** Method which enables clients to look up a bean or a collection of beans in a globally available directory.

**File Cache** The file cache contains information about files and static file content. The file cache is turned on by default.

**firewall** an electronic boundary that allows a network administrator to restrict the flow of information across networks in order to enforce security.

**form action handler** A specially defined method in servlet or application logic that performs an action based on a named button on a form.

**FQDN** Fully Qualified Domain Name. The full name of a system, containing its hostname and its domain name.

**general ACL** A named list in the Sun Java System Directory Server that relates a user or group with one or more permissions. This list can be defined and accessed arbitrarily to record any set of permissions.

**generic servlet** A servlet that extends `javax.servlet.GenericServlet`. Generic servlets are protocol-independent, meaning that they contain no inherent support for HTTP or any other transport protocol. Contrast with [HTTP servlet](#).

**global database connection** A database connection available to multiple components. Requires a resource manager.

**global transaction** A transaction that is managed and coordinated by a transaction manager and can span multiple databases and processes. The transaction manager typically uses the XA protocol to interact with the database backends. *See* [local transaction](#).

**granularity level** The approach to dividing an application into pieces. A *high level of granularity* means that the application is divided into many smaller, more narrowly defined Enterprise JavaBeans (EJBs). A *low level of granularity* means the application is divided into fewer pieces, producing a larger program.

**group** A group of users that are related in some way. Group membership is usually maintained by a local system administrator. *See* [user](#), [role](#).

**HADB** *See* [High-Availability Database \(HADB\)](#).

**HADB node** A set of HADB processes, a dedicated area of shared memory, and one or more secondary storage devices used for storing and updating session data. Each active (data storage) node must have a mirror node; therefore nodes occur in pairs. In addition, two or more spare nodes can be included to maximize availability. If an active node fails and cannot recover within a timeout period, the spare node copies the data from the mirror node and becomes active. *See also* [High-Availability Database \(HADB\)](#), [active node](#), [spare node](#), [mirror node](#), and [data redundancy unit \(DRU\)](#).

**handle** An object that identifies an enterprise bean. A client may serialize the handle, and then later deserialize it to obtain a reference to the bean.

**health check** The process of sending periodic requests by the load balancer to an unhealthy application server instance to determine if it has become healthy again. *See also* [healthy application server instance](#), [unhealthy application server instance](#).

**healthy application server instance** An application server instance in a cluster that is responding normally to load balancer requests. *See also* [unhealthy application server instance](#).

**Heuristic Decision** The transactional mode used by a particular transaction. A transaction has to either Commit or Rollback.

**high availability** The capability of a system to provide continuous or near continuous operations despite the failure of one or more parts of the system.

**High-Availability Database (HADB)** A highly scalable, highly available session state persistence infrastructure. The Sun Java System Application Server uses the HADB to store HTTP session states and stateful session bean states. *See also* [HADB node](#), [active node](#), [spare node](#), [mirror node](#), and [data redundancy unit \(DRU\)](#).

**home interface** A mechanism that defines the methods that enable a client to create and remove an enterprise bean.

**host-IP authentication** A security mechanism used for of limiting access to the Administration Server, or the files and directories on a web site by making them available only to clients using specific computers.

**HTML** Hypertext Markup Language. A coding markup language used to create documents that can be displayed by web browsers. Each block of text is surrounded by codes that indicate the nature of the text.

**HTML page** A page coded in HTML and intended for display in a web browser.

**HTTP** Hypertext Transfer Protocol. The Internet protocol that fetches hypertext objects from remote hosts. It is based on TCP/IP.

**HTTP servlet** A servlet that extends `javax.servlet.HttpServlet`. These servlets have built-in support for the HTTP protocol. Contrast with [generic servlet](#).

**HTTPS** HyperText Transmission Protocol, Secure. HTTP for secure transactions.

**IDE** Integrated Development Environment. Software that allows you to create, assemble, deploy, and debug code from a single, easy-to-use interface.

**IIOP** Internet Inter-ORB Protocol. Transport-level protocol used by both Remote Method Invocation (RMI) over IIOP and Common Object Request Broker Architecture (CORBA).

**IIOP endpoint** An [IIOP listener](#) that has been configured for an IIOP cluster to enable high availability of RMI/IIOP requests.

**IIOP cluster** A [cluster](#) that has been configured for high availability of RMI/IIOP requests.

**IIOP listener** The IIOP listener is a listen socket that listens on a specified port and accepts incoming connections from CORBA based client application.

**IMAP** Internet Message Access Protocol.

**instance disabling (from cluster)** The process by which an application server instance in a cluster is given enough time to serve the existing requests that it is handling before the load balancer stops sending requests to it. *See also* [instance enabling \(in cluster\)](#), [quiescing](#).

**instance enabling (in cluster)** The process of adding an application server instance to a cluster so the load balancer starts sending requests to the application server instance. *See also* [instance disabling \(from cluster\)](#).

**IP address** A structured, numeric identifier for a computer or other device on a TCP/IP network. The format of an IP address is a 32-bit numeric address written as four numbers separated by periods. Each number can be zero to 255. For example, 123.231.32.2 could be an IP address.

**isolation level** *See* [transaction isolation level](#).

**J2EE** Java 2 Enterprise Edition. An environment for developing and deploying multi-tiered, web-based enterprise applications. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing these applications.

**JAF** The JavaBeans Activation Framework (JAF) integrates support for MIME data types into the Java platform. *See* [MIME Data Type](#).

**JAR file** Java ARchive file. A file used for aggregating many files into one file. JAR files have the .jar extension.

**JAR file contract** Java ARchive contract that specifies what information must be in the enterprise bean package.

**JAR file format** Java ARchive file format. A platform-independent file format that aggregates many files into one file. Multiple applets and their requisite components (class files, images, sounds, and other resource files) can be bundled in a JAR file and subsequently downloaded to a browser in a single HTTP transaction. The JAR files format also supports file compression and digital signatures.

**JavaBean** A portable, platform-independent reusable component model.

**Java IDL** Java Interface Definition Language. APIs written in the Java programming language that provide a standards-based compatibility and connectivity with Common Object Request Broker Architecture (CORBA).

**JavaMail session** An object used by an application to interact with a mail store. Application code uses the JNDI Service to locate JavaMail session resources objects using a JNDI name.

**JAXM** Java API for XML Messaging. Enables applications to send and receive document-oriented XML messages using the SOAP standard. These messages can be with or without attachments.

**JAXP** Java API for XML Processing. A Java API that supports processing of XML documents using DOM, SAX, and XSLT. Enables applications to parse and transform XML documents independent of a particular XML processing implementation.

**JAXR** Java API for XML Registry. Provides a uniform and standard Java API for accessing different kinds of XML registries. Enables users to build, deploy and discover web services.

**JAX-RPC** Java API for XML-based Remote Procedure Calls. Enables developers to build interoperable web applications and web services based on XML-based RPC protocols.

**JDBC** Java Database Connectivity. A standards-based set of classes and interfaces that enable developers to create data-aware components. JDBC implements methods for connecting to and interacting with data sources in a platform- and vendor-independent way.

**JDBC connection pool** A pool that combines the JDBC data source properties used to specify a connection to a database with the connection pool properties.

**JDBC resource** A resource used to connect an application running within the application server to a database using an existing JDBC connection pool. Consists of a JNDI name (which is used by the application) and the name of an existing JDBC connection pool.

**JDK** Java Development Kit. The software that includes the APIs and tools that developers need to build applications for those versions of the Java platform that preceded the Java 2 Platform. *See also* [JDK](#).

**JMS** Java Message Service. A standard set of interfaces and semantics that define how a JMS client accesses the facilities of a JMS message service. These interfaces provide a standard way for Java programs to create, send, receive, and read messages.



**JMS-administered object** A pre-configured JMS object—a connection factory or a destination—created by an administrator for use by one or more JMS clients. The use of administered objects allows JMS clients to be provider-independent; that is, it isolates them from the proprietary aspects of a provider. These objects are placed in a JNDI name space by an administrator and are accessed by JMS clients using JNDI lookups.

**JMS client** An application (or software component) that interacts with other JMS clients using a JMS message service to exchange messages.

**JMS connection factory** The JMS administered object a JMS client uses to create a connection to a JMS message service.

**JMS destination** The physical destination in a JMS message service to which produced messages are delivered for routing and subsequent delivery to consumers. This physical destination is identified and encapsulated by an JMS administered object that a JMS client uses to specify the destination for which it is producing messages and/or from which it is consuming messages.

**JMS messages** Asynchronous requests, reports, or events that are consumed by JMS clients. A message has a header (to which additional fields can be added) and a body. The message header specifies standard fields and optional properties. The message body contains the data that is being transmitted.

**JMS provider** A product that implements the JMS interfaces for a messaging system and adds the administrative and control functions needed for a complete product.

**JMS Service** Software that provides delivery services for a JMS messaging system, including connections to JMS clients, message routing and delivery, persistence, security, and logging. The message service maintains physical destinations to which JMS clients send messages, and from which the messages are delivered to consuming clients.

**JNDI** Java Naming and Directory Interface. This is a standard extension to the Java platform, providing Java technology-enabled applications with a unified interface to multiple naming and directory services in the enterprise. As part of the Java Enterprise API set, JNDI enables seamless connectivity to heterogeneous enterprise naming and directory services.

**JNDI name** A name used to access a resource that has been registered in the JNDI naming service.

**JRE** Java Runtime Environment. A subset of the Java Development Kit (JDK) consisting of the Java virtual machine, the Java core classes, and supporting files that provides runtime support for applications written in the Java programming language. *See also* [JDK](#).

**JSP** JavaServer Page. A text page written using a combination of HTML or XML tags, JSP tags, and Java code. JSPs combine the layout capabilities of a standard browser page with the power of a programming language.

**jspc utility** The compiler for JSPs. It checks all JSPs for compliance with the JSP specification.

**JTA** Java Transaction API. An API that allows applications and J2EE servers to access transactions.

**JTS** Java Transaction Service. The Java service for processing transactions.

**key-pair file** *See* [trust database](#).

**LDAP** Lightweight Directory Access Protocol. LDAP is an open directory access protocol that runs over TCP/IP. It is scalable to a global size and millions of entries. Using Sun Java System Directory Server, a provided LDAP server, you can store all of your enterprise's information in a single, centralized repository of directory information that applications can access through the network.

**LDIF** LDAP Data Interchange Format. Format used to represent Sun Java System Directory Server entries in text form.

**lifecycle event** A stage in the server life cycle, such as startup or shutdown.

**lifecycle module** A module that listens for and performs its tasks in response to events in the server life cycle.

**Listener** A class, registered with a posting object, that says what to do when an event occurs.

**load balancer** A component that routes requests to application server instances, detects failure of application server instances, and helps implement high availability.

**load balancer polling** A periodic check by the load balancer to determine if the configuration settings for the load balancer (as specified in the `loadbalancer.xml` file) have changed. The load balancer checks the time stamp of the `loadbalancer.xml` file to ascertain if the file has changed. If changes have occurred, the entire configuration is reloaded as specified in the `loadbalancer.xml` file. Polling is done only if dynamic reconfiguration is enabled. *See also* [dynamic reconfiguration](#).

**load balancer plug-in** *See* [load balancer](#).

**loadbalancer.xml file** The configuration file in which the configuration settings for the load balancer are defined.

**local database connection** The transaction context in a local connection is local to the current process and to the current data source, not distributed across processes or across data sources.

**local interface** An interface that provides a mechanism for a client that is located in the same Java Virtual Machine (JVM) with a session or entity bean to access that bean.

**local session** A user session that is only visible to one server.

**local transaction** A transaction that is native to one database and is restricted within a single process. Local transactions work only against a single backend. Local transactions are typically demarcated using JDBC APIs. *See also* [global transaction](#).

**mapping** The ability to tie an object-oriented model to a relational model of data, usually the schema of a relational database. The process of converting a schema to a different structure. Also refers to the mapping of users to security roles.

**MDB** *See* [message-driven bean](#).

**message-driven bean** An enterprise bean that is an asynchronous message consumer. A message-driven bean has no state for a specific client, but its instance variables may contain state across the handling of client messages, including an open database connection and an object reference to an EJB object. A client accesses a message-driven bean by sending messages to the destination for which the message-driven bean is a message listener.

**messaging** A system of asynchronous requests, reports, or events used by enterprise applications that allows loosely coupled applications to transfer information reliably and securely.

**metadata** Information about a component, such as its name, and specifications for its behavior.

**management information base (MIB)** A tree-like structure that defines the variables the master SNMP agent can access. The MIB provides access to the HTTP server's network configuration, status, and statistics. Using SNMP, you can view this information from the network management workstation (NMS). *See also* [network management station \(NMS\)](#) and [SNMP](#).

**MIME Data Type** MIME (Multi-purpose Internet Mail Extension) types control what types of multimedia files your system supports.

**mirror node** An active HADB node that contains the same data as another active node, but resides in the other data redundancy unit. Each active node must have a mirror node; therefore nodes occur in pairs. When a node detects that its mirror node has failed, it takes over the failed node's role and continues service. *See also* [HADB node](#), [active node](#), [spare node](#), and [data redundancy unit \(DRU\)](#).

**module** A web application, enterprise bean, message-driven bean, application client, or connector that has been deployed individually, outside an application. *See also* [application](#), [component](#), [lifecycle module](#).

**network management station (NMS)** A machine used to remotely manage a specific network. Usually, the NMS software will provide a graph to display collected data or use that data to make sure the server is operating within a particular tolerance. *See also* [SNMP](#).

**NTV** Name, Type, Value.

**object persistence** *See* [persistence](#).

**O/R mapping tool** Object-to-relational [database] tool. A mapping tool within the Sun Java System Application Server Administrative interface that creates XML deployment descriptors for entity beans.

**package** A collection of related classes that are stored in a common directory. They are often literally packaged together in a Java archive JAR file. *See also* [assembly](#), [deployment](#).

**parameter** A name/value pair sent from the client, including form field data, HTTP header information, and so on, and encapsulated in a request object. Contrast with attribute. More generally, an argument to a Java method or database-prepared command.

**passivation** A method of releasing a bean's resources from memory without destroying the bean. In this way, a bean is made to be persistent, and can be recalled without the overhead of instantiation.

**permission** A set of privileges granted or denied to a user or group. *See also* [ACL](#).

**persistence** For enterprise beans, the protocol for transferring the state of an entity bean between its instance variables and an underlying database. Opposite of [transience](#). For sessions, the session storage mechanism.

**persistence frequency** A setting for an application server instance or an application that specifies the frequency at which the HTTP session state is stored in the High Availability Database (HADB). The session can be stored after every web-request (called the web-method frequency mode), or after a configurable time interval (called the time-based frequency mode). The persistence frequency can be set only if the session state is stored in the HADB.

**persistence manager** The entity responsible for the persistence of the entity beans installed in the container.

**persistence scope** A setting for an application server instance or a web application that specifies how much of a session is stored. The available options are: saving an entire session, saving the modified attributes of the session, and saving the session only if it has been changed since it was stored last. The persistence scope can be set only if the session state is stored in the HADB.

**persistence type** A setting for an application server instance or a web application that defines where and how the session state will be stored. The persistence types available are `memory`, `file`, and `ha`.

**persistent state** Where the state of an object is kept in persistent storage, usually a database.

**pluggable authentication** A mechanism that allows J2EE applications to use the Java Authentication and Authorization Service (JAAS) feature from the J2SE platform. Developers can plug in their own authentication mechanisms.

**point-to-point delivery model** Producers address messages to specific queues; consumers extract messages from queues established to hold their messages. A message is delivered to only one message consumer.

**pooling** The process of providing a number of preconfigured resources to improve performance. If a resource is pooled, a component can use an existing instance from the pool rather than instantiating a new one. In the Sun Java System Application Server, database connections, servlet instances, and enterprise bean instances can all be pooled.

**POP3** Post Office Protocol

**prepared command** A database command (in SQL) that is precompiled to make repeated execution more efficient. Prepared commands can contain parameters. A prepared statement contains one or more prepared commands.

**prepared statement** A class that encapsulates a `QUERY`, `UPDATE`, or `INSERT` statement that is used repeatedly to fetch data. A prepared statement contains one or more prepared commands.

**presentation layout** The format of web page content.

**presentation logic** Activities that create a page in an application, including processing a request, generating content in response, and formatting the page for the client. Usually handled by a web application.

**primary key** The unique identifier that enables the client to locate a particular entity bean.

**primary key class name** A variable that specifies the fully qualified class name of a bean's primary key. Used for JNDI lookups.

**principal** The identity assigned to an entity as a result of authentication.

**private key** See [public key cryptography](#).

**process** Execution sequence of an active program. A process is made up of one or more threads.

**programmatic security** The process of controlling security explicitly in code rather than allowing the component's container (for instance, a bean's container or a servlet engine) to handle it. Opposite of [declarative security](#).

**programmer-demarcated transaction** See [bean-managed transaction](#).

**property** A single attribute that defines the behavior of an application component. In the `server.xml` file, a property is an element that contains a name/value pair.

**public key cryptography** A form of cryptography in which each user has a public key and a private key. Messages are sent encrypted with the receiver's public key; the receiver decrypts them using the private key. Using this method, the private key never has to be revealed to anyone other than the user.

**publish/subscribe delivery model** Publishers and subscribers are generally anonymous and may dynamically publish or subscribe to a topic. The system distributes messages arriving from a topic's multiple publishers to its multiple subscribers.

**QOS** QOS (Quality of Service) refers to the performance limits you set for a server instance or virtual server. For example, if you are an ISP, you might want to charge different amounts of money for virtual servers depending on how much bandwidth is provided. You can limit two areas: the amount of bandwidth and the number of connections.

**queue** An object created by an administrator to implement the point-to-point delivery model. A queue is always available to hold messages even when the client that consumes its messages is inactive. A queue is used as an intermediary holding place between producers and consumers.

**quiescing** The process of shutting down an application server instance or a web application in a phased manner. The quiescing period can be specified for each application server instance and each web application. See also [quiescing period](#).

**quiescing period** The time interval after which the load balancer stops sending sticky requests to the application server instance or web application that is being quiesced. See also [quiescing](#), [assigned request](#).

**RAR file** Resource ARchive. A JAR archive that contains a resource adapter.

**RDB** Relational database.

**RDBMS** Relational database management system.

**read-only bean** An entity bean that is never modified by an EJB client. See also [entity bean](#).

**realm** A scope over which a common security policy is defined and enforced by the security administrator of the security service. Also called a *security policy domain* or *security domain* in the J2EE specification.

**remote interface** One of two interfaces for an Enterprise JavaBean. The remote interface defines the business methods callable by a client.

**request object** An object that contains page and session data produced by a client, passed as an input parameter to a servlet or JavaServer Page (JSP).

**resource manager** An object that acts as a facilitator between a resource such as a database or message broker, and client(s) of the resource such as Sun Java System Application Server processes. Controls globally-available data sources.

**resource reference** An element in a deployment descriptor that identifies the component's coded name for the resource.

**response object** An object that references the calling client and provides methods for generating output for the client.

**ResultSet** An object that implements the `java.sql.ResultSet` interface. `ResultSet`s are used to encapsulate a set of rows retrieved from a database or other source of tabular data.

**reusable component** A component created so that it can be used in more than one capacity, for instance, by more than one resource or application.

**RMI** Remote Method Invocation. A Java standard set of APIs that enable developers to write remote interfaces that can pass objects to remote processes.

**RMIC** Remote Method Invocation Compiler.

**role** A functional grouping of subjects in an application, represented by one or more groups in a deployed environment. *See also* [user](#), [group](#).

**rollback** Cancellation of a transaction.

**row** A single data record that contains values for each column in a table.

**RowSet** An object that encapsulates a set of rows retrieved from a database or other source of tabular data. `RowSet` extends the `java.sql.ResultSet` interface, enabling `ResultSet` to act as a JavaBeans component.



**RPC** Remote Procedure Call. A mechanism for accessing a remote object or service.

**runtime system** The software environment in which programs run. The runtime system includes all the code necessary to load programs written in the Java programming language, dynamically link native methods, manage memory, and handle exceptions. An implementation of the Java virtual machine is included, which may be a Java interpreter.

**SAF** Server Application Function. A function that participates in request processing and other server activities

**schema** The structure of the underlying database, including the names of tables, the names and types of columns, index information, and relationship (primary and foreign key) information.

**Secure Socket Layer** See [SSL](#).

**security** A screening mechanism that ensures that application resources are only accessed by authorized clients.

**serializable object** An object that can be deconstructed and reconstructed, which enables it to be stored or distributed among multiple servers.

**server instance** A Sun Java System Application Server can contain multiple instances in the same installation on the same machine. Each instance has its own directory structure, configuration, and deployed applications. Each instance can also contain multiple virtual servers. See also [virtual server](#).

**servlet** An instance of the `Servlet` class. A servlet is a reusable application that runs on a server. In the Sun Java System Application Server, a servlet acts as the central dispatcher for each interaction in an application by performing presentation logic, invoking business logic, and invoking or performing presentation layout.

**servlet engine** An internal object that handles all servlet metafunctions. Collectively, a set of processes that provide services for a servlet, including instantiation and execution.

**servlet runner** The part of the servlet engine that invokes a servlet with a request object and a response object. See [servlet engine](#).

**session** An object used by a servlet to track a user's interaction with a web application across multiple HTTP requests.

**session bean** An enterprise bean that is created by a client; usually exists only for the duration of a single client-server session. A session bean performs operations for the client, such as calculations or accessing other EJBs. While a session bean may be transactional, it is not recoverable if a system crash occurs. Session bean objects can be either stateless (not associated with a particular client) or stateful (associated with a particular client), that is, they can maintain conversational state across methods and transactions. *See also* [stateful session bean](#), [stateless session bean](#).

**session cookie** A cookie that is returned to the client containing a user session identifier. *See also* [sticky cookie](#).

**session persistence** *See* [persistence](#).

**session timeout** A specified duration after which the Sun Java System Application Server can invalidate a user session. *See* [session](#).

**single sign-on** A situation where a user's authentication state can be shared across multiple J2EE applications in a single virtual server instance.

**SMTP** Simple Mail Transport Protocol

**SNMP** SNMP (Simple Network Management Protocol) is a protocol used to exchange data about network activity. With SNMP, data travels between a managed device and a network management station (NMS). A managed device is anything that runs SNMP: hosts, routers, your web server, and other servers on your network. The NMS is a machine used to remotely manage that network.

**SOAP** The Simple Object Access Protocol (SOAP) uses a combination of XML-based data structuring and Hyper Text Transfer Protocol (HTTP) to define a standardized way of invoking methods in objects distributed in diverse operating environments across the Internet.

**spare node** An HADB node that can replace a failed active node. If an active node fails, a spare node copies data from the mirror node and becomes active. *See also* [HADB node](#), [active node](#), [mirror node](#), and [data redundancy unit \(DRU\)](#).

**SQL** Structured Query Language. A language commonly used in relational database applications. SQL2 and SQL3 designate versions of the language.

**SSL** Secure Sockets Layer. A protocol designed to provide secure communications on the Internet.

**state** 1. The circumstances or condition of an entity at any given time. 2. A distributed data storage mechanism which you can use to store the state of an application using the Sun Java System Application Server feature interface `IState2`. *See also* [conversational state](#), [persistent state](#).

**stateful session bean** A session bean that represents a session with a particular client and which automatically maintains state across multiple client-invoked methods.

**stateless session bean** A session bean that represents a stateless service. A stateless session bean is completely transient and encapsulates a temporary piece of business logic needed by a specific client for a limited time span.

**sticky cookie** A cookie that is returned to the client to force it to always connect to the same server process. *See also* [session cookie](#).

**stored procedure** A block of statements written in SQL and stored in a database. You can use stored procedures to perform any type of database operation, such as modifying, inserting, or deleting records. The use of stored procedures improves database performance by reducing the amount of information that is sent over a network.

**streaming** A technique for managing how data is communicated through HTTP. When results are streamed, the first portion of the data is available for use immediately. When results are not streamed, the whole result must be received before any part of it can be used. Streaming provides a way to allow large amounts of data to be returned in a more efficient way, improving the perceived performance of the application.

**Sun Java System Directory Server** The Sun Java System version of Lightweight Directory Access Protocol (LDAP). Every instance of Sun Java System Application Server uses Sun Java System Directory Server to store shared server information, including information about users and groups. *See also* [LDAP](#).

**Sun Java System Message Queue** The Sun Java System enterprise messaging system that implements the Java Message Service (JMS) open standard: it is a JMS provider.

**system administrator** The person who administers Sun Java System Application Server software and deploys Sun Java System Application Server applications.

**table** A named group of related data in rows and columns in a database.

**thread** An execution sequence inside a process. A process may allow many simultaneous threads, in which case it is multi-threaded. If a process executes each thread sequentially, it is single-threaded.

**TLS** Transport Layer Security. A protocol that provides encryption and certification at the transport layer, so that data can flow through a secure channel without requiring significant changes to the client and server applications.

**topic** An object created by an administrator to implement the publish/subscribe delivery model. A topic may be viewed as node in a content hierarchy that is responsible for gathering and distributing messages addressed to it. By using a topic as an intermediary, message publishers are kept separate from message subscribers.

**transaction** A set of database commands that succeed or fail as a group. All the commands involved must succeed for the entire transaction to succeed.

**Transaction Attribute** A transaction attribute controls the scope of a transaction.

**transaction context** A transaction's scope, either local or global. See [local transaction](#), [global transaction](#).

**transaction isolation level** Determines the extent to which concurrent transactions on a database are visible to one-another.

**transaction manager** An object that controls a global transaction, normally using the XA protocol. See [global transaction](#).

**Transaction Recovery** Automatic or manual recovery of distributed transactions.

**transience** A protocol that releases a resource when it is not being used. Opposite of [persistence](#).

**trust database** A security file that contains the public and private keys; also referred to as the [key-pair file](#).

**UDDI** Universal Description, Discovery, and Integration. Provides worldwide registry of web services for discovery and integration.

**unassigned request** A request for a new session.

**unhealthy application server instance** An application server instance in a cluster that is not responding to requests from the load balancer.

**URI** Universal Resource Identifier. Describes a specific resource at a domain. Locally described as a subset of a base directory, so that `/ham/burger` is the base directory and a URI specifies `toppings/cheese.html`. A corresponding URL would be `http://domain:port/toppings/cheese.html`.

**URL** Uniform Resource Locator. An address that uniquely identifies an HTML page or other resource. A web browser uses URLs to specify which pages to display. A URL describes a transport protocol (for example, HTTP, FTP), a domain (for example, `www.my-domain.com`), and optionally a URI.

**user** A person who uses an application. Programmatically, a user consists of a user name, password, and set of attributes that enables an application to recognize a client. *See also* [group](#), [role](#).

**user session** A series of user application interactions that are tracked by the server. Sessions maintain user state, persistent objects, and identity authentication.

**versioning** *See* [dynamic reloading](#).

**virtual server** A virtual web server that serves content targeted for a specific URL. Multiple virtual servers may serve content using the same or different host names, port numbers, or IP addresses. The HTTP service can direct incoming web requests to different virtual servers based on the URL. Also called a virtual host. A web application can be assigned to a specific virtual server. A server instance can have multiple virtual servers. *See also* [server instance](#).

**WAR file** Web ARchive. A Java archive that contains a web module. WAR files have the `.war` extension.

**web application** A collection of servlets, JavaServer Pages, HTML documents, and other web resources, which might include image files, compressed archives, and other data. A web application may be packaged into an archive (a WAR file) or exist in an open directory structure. Sun Java System Application Server also supports some non-Java web application technologies, such as SHTML and CGI.

**web cache** A Sun Java System Application Server feature that enables a servlet or JSP to cache its results for a specific duration in order to improve performance. Subsequent calls to that servlet or JSP within the duration are given the cached results so that the servlet or JSP does not have to execute again.

**web connector plug-in** An extension to a web server that enables it to communicate with the Sun Java System Application Server.

**web container** See [container](#).

**web module** An individually deployed web application. See [web application](#).

**web server** A host that stores and manages HTML pages and web applications, but not full J2EE applications. The web server responds to user requests from HTTP clients.

**Web Server Plug-in** The web server plug-in is an HTTP reverse proxy plug-in that allows you to instruct a Sun Java System Web Server or Sun Java System Application Server to forward certain HTTP requests to another server.

**web service** A service offered via the web. A self-contained, self-describing, modular application that can accept a request from a system across the Internet or an intranet, process it, and return a response.

**WSDL** Web Service Description Language. An XML-based language used to define web services in a standardized way. It essentially describes three fundamental properties of a web service: definition of the web service, how to access that web service, and the location of that web service.

**XA protocol** A database industry standard protocol for distributed transactions.

**XML** Extensible Markup Language. A language that uses HTML-style tags to identify the kinds of information used in documents as well as to format documents.

# Index

## A

- about HTTPS routing [422](#)
- accept language header, parsing [375](#)
- acceptor threads
  - specifying number via HTTP listener [59](#)
  - virtual servers [365](#)
- access [119](#)
- access control, using virtual servers [370](#)
- access log files [107](#), [118](#)
  - configuring [119](#)
  - rotation [107](#)
  - viewing [118](#)
- access.log file [94](#)
- ACL, attributes [605](#)
- acl, dotted name [605](#)
- activation, definition [187](#)
- adaptor, resource [201](#)
- add-iiop-cluster-endpoint command [579](#)
- additional document directories [390](#)
- AddLog [314](#)
- add-resources command [301](#), [579](#)
- administered objects *See* JMS administered objects
- Administration interface
  - accessing [45](#), [52](#)
  - administering transactions [214](#)
  - automatic transaction recovery [205](#)
  - configuring log service attributes [115](#)
  - general settings, configuring [86](#)
  - JVM options, configuring [87](#)
  - JVM Profiler, configuring [88](#)
  - online help, accessing [47](#)
  - path settings, configuring [87](#)
  - shutting down the Administration Server [55](#)
  - standard buttons [47](#)
  - tabs, using [46](#)
  - using [45](#)
- Administration Server
  - about [51](#)
  - applying changes [58](#)
  - control settings, viewing [57](#)
  - restarting manually (Unix) [53](#)
  - security auditing [129](#)
  - settings, accessing [57](#)
  - shutting down, methods for [54](#)
  - starting the SNMP master agent [177](#)
  - starting, methods for [52](#)
  - stopping [56](#)
  - stopping manually [55](#)
- administration, tools and associated functions [44](#)
- administrative domains *See* domains
- admin-service [104](#)
- afterBegin [211](#)
- afterCompletion [211](#), [212](#)
- agents, SNMP [170](#)
- ALERT [102](#)
- analyzer, log
  - running [120](#)
- ansi\_x3.4-1968 [399](#)
- ansi\_x3.4-1986 [399](#)
- Ant tasks [560](#)
- Apache Web Server, configuring [329](#)
- appclient utility [560](#)
- Application Client Container (ACC)

- client side logging 105
- application client JAR file 334
- application log output, redirecting 106
- Application Server
  - logging features and functions 93
  - overview and key features 43
  - request handling 312
- application server instances, *See* instances, application server
- application, dotted name 600
- application.xml deployment descriptor 336
- application-client.xml deployment descriptor 336
- applications
  - attributes 600
  - connection sharing 260
  - deploying on cluster 450
  - directory structure 338
  - disabling 344
  - disabling in cluster 451, 458
  - dynamic reloading 344
  - elements of web 181
  - enabling in cluster 451
  - environment entries 225
  - J2EE, introduction 335
  - JMS and 279
  - JNDI lookup names 338
  - naming standards 337
  - quiescing in cluster 458
  - redeploying on cluster 452
  - resource environment references 235
  - resource references 234
  - runtime environment 340
  - specifying quiescing period for 451
  - undeploying from cluster 461
- appserv.mib 163
  - managed objects and descriptions 164
- appservd 81
- appservd-wdog.exe 81
- appserv-wdog 81
- archiving, log files 106
- AS\_ADMIN\_HOST 567
- AS\_ADMIN\_INSTANCE 567
- AS\_ADMIN\_PASSWORD 567
- AS\_ADMIN\_PORT 567
- AS\_ADMIN\_PREFIX 573
- AS\_ADMIN\_SECURE 567
- AS\_ADMIN\_USER 567
- asadmin utility
  - about 560
  - attributes 584
  - command syntax 562
  - commands 562
  - concurrent access 578
  - database, administering and monitoring transactions 216
  - default values 611
  - dotted names 584
  - environment commands 565
  - environment variables 611
  - escape characters 570
  - exit status 576
  - export 565
  - extracting monitoring data 135
  - from command line 569
  - from pipe 570
  - from script 570
  - get 572
  - help 575
  - interactive 565
  - JVM settings 88
  - license commands 50
  - local 568
  - long options 611
  - multimode 564
  - non-interactive 565
  - operands 563
  - options 562
  - password file option 567
  - reconfig 573
  - remote 568
  - restarting instances 80
  - security 578
  - set 572
  - short options 611
  - singlemode 564
  - starting and stopping instances 74
  - transaction administration 216
  - unset 566
  - usage 578
- ascii 399
- atomicity 200



- attributes
  - about 639
  - asadmin 584
  - EJB container (that can be monitored) 193
  - transaction, deployment descriptors 209
  - transactions 206
  - virtual server 181
  - web-module 182
- auth-db 607
- authentication 305
- authentication database attributes 607
- authorization realm attributes 608
- auth-passthrough 315, 316
- authrealm 608
- AuthTrans 314
- AuthTrans qos-handler 159
- AuthTrans-class 316
- auto-commit connection validation 259
- availability
  - EJB container configuration 196
  - enabling and disabling 474
  - web container 185
- availability features, in deployment 345
- avax.transaction.UserTransaction 212

## B

- bean-cache
  - monitoring attribute names 145
- beanIdleTimeoutInSeconds 194
- bean-managed transactions
  - not allowed for entity beans 213
- bean-method
  - monitoring attribute names 146
- beans, message-driven, characteristics 192
- beforeCompletion 211
- boolean options 562
- BSM auditing on Solaris 134
- business methods, transactions 209, 211

## C

- cache control directives, setting 403
- cache settings, configuring EJB 195
- CacheBucket
  - monitoring attributes 152
  - monitoring HTTP server elements 148
- cacheDatabaseMetaData property 524
- cacheFaultsPercentage 194
- cache-hits 146
- cache-misses 145
- cache-resize-quantity 145
- capture-schema utility 560
- CGIs 397
  - settings for virtual servers 377
  - with virtual servers 371
- character set
  - changing 398
  - iso\_8859-1 399
  - us-ascii 399
- check-passthrough 318
- checkpointing 480
- chroot settings 377
- ciphers, TLS Rollback option 309
- cladmin command
  - configuring session persistence 485
  - log files 655
  - running 653
  - syntax 653
  - using while running multiple clusters 655
- classpathprefix 89
- clear command 579
- clearhistory command 579
- clear-session-store command 470, 579
- client name mapping example 138
- clients
  - lists of accesses 119
  - requests 312
- clinstance.conf file
  - entries in 651
  - format 651
  - sample file 652
- clpassword.conf file 652
- clsetup command 464, 468

- cluster element 642
- clusters
  - about 443
  - and the session store 472
  - defining 445
  - deploying web application on 450
  - disabling web application in 451
  - enabling web application in 451
  - reconfiguring application server instance in running cluster 461
  - redeploying web applications on 452
  - removing 461
  - removing application server instance from 453
  - requirements for adding application server instances to 449
  - sample configuration 455
  - session persistence for 485
  - specifying name for session persistence configuration 472
  - starting 450
  - stopping 454
  - stopping application server instance in 453
  - time interval for health check 446
  - timeout value for health check 446
  - undeploying web applications from 461
  - URL for health checker 446
  - using multiple clusters for online upgrade 460
- command line, asadmin from 569
- command-line interface
  - name mapping, monitoring 138
  - shutting down the Administration Server 56
  - starting the Administration Server 53
- commands
  - asadmin 562
  - license 50
- commits *See* transactions, commits
- community string, with SNMP agent 168
- compiling Apache 330
- components, MDB, *See* MDB
- concurrent access, asadmin 578
- concurrent connections
  - virtual servers, quality of service 161
- CONFIG 102, 171, 173, 174
  - master agent, editing 174
- ConfigPath attribute 548
- configpath option 517
- configuration files, about 49
- configure-session-persistence command 484, 579
- connection factories
  - defined 230
  - JNDI 230
  - URL 235
- Connection object 230
- connection pool
  - properties for HADB 523
  - settings for HADB 523
- connection pooling
  - about 258
  - datasource object 245
- connection sharing 260
- connection validation 259
- Connection Validation Required setting 523
- ConnectionQueue 153
  - monitoring attributes 149
  - monitoring ConnectionQueueBucket attributes 150
  - monitoring HTTP server elements 147
- ConnectionQueueBucket
  - monitoring attributes 150
  - monitoring HTTP server elements 147
- connections, shareable or non-shareable 205
- ConnectionTrace attribute 548
- connector modules
  - attributes 603
  - deployment directory structure 339
- connector-module 603
- container-managed transactions 205
- containers
  - EJB, responsibilities 188
  - MDB 280
  - Web, about 179
- context-root 182
- context-root attribute 644
- control settings, viewing Administration Server 57
- conversational state 189
- CORBA, about 303
- CoreFile attribute 548
- COSNaming service 229
- Count200 through Count503 154

- Count2xx through Count5xx [154](#)
- CountAsyncAddrLookups [151](#)
- CountAsyncLookupsInProgress [151](#)
- CountAsyncNameLookups [151](#)
- CountBytesReceived [154](#)
- CountBytesTransmitted [154](#)
- CountCacheEntries [151](#)
- CountCacheHits [151](#)
- CountCacheMisses [151](#)
- Countcalls [155](#)
- CountConfigurations
  - monitoring Process attributes [150](#)
- CountConnections [152](#)
- CountContentHits [153](#)
- CountContentMisses [153](#)
- CountEntries [152](#)
- CountFlushes [152](#)
- CountHits [152](#), [153](#)
- CountInfoHits [153](#)
- CountInfoMisses [153](#)
- CountMisses [153](#)
- CountOpenConnections [154](#)
- CountOpenEntries [152](#)
- CountOther [154](#)
- CountOverflow
  - monitoring ConnectionQueueBucket
    - attributes [150](#)
- CountQueued [151](#)
  - monitoring ConnectionQueueBucket
    - attributes [150](#)
- CountRefusals [152](#)
- CountRequests [154](#), [155](#)
- CountThreads [151](#)
- CountThreadsIdle [151](#)
- CountTimeouts [152](#)
- CountTotalConnection
  - monitoring ConnectionQueueBucket
    - attributes [150](#)
- CountTotalQueued
  - monitoring ConnectionQueueBucket
    - attributes [150](#)
- cp367 [399](#)
- cp819 [399](#)

- create-acl command [580](#)
- create-authdb command [580](#)
- create-auth-realm command [580](#)
- create-custom-resource command [580](#)
- create-domain command [63](#), [580](#)
- create-file-user command [580](#)
- create-http-listener command [372](#), [580](#)
- create-http-qos command [358](#), [378](#), [580](#)
- create-iiop-listener command [580](#)
- create-instance command [82](#), [580](#)
- create-javamail-resource command [580](#)
- create-jdbc-connection-pool command [255](#), [580](#)
- create-jdbc-resource command [246](#), [258](#), [580](#)
- create-jmsdest command [301](#), [580](#)
- create-jms-resource command [301](#), [580](#)
- create-jndi-resource command [580](#)
- create-jvm-options command [89](#), [580](#)
- create-lifecycle-module command [349](#), [580](#)
- create-mime command [361](#), [580](#)
- create-persistence-resource command [580](#)
- create-profiler command [580](#)
- create-session-store command [468](#), [580](#)
  - environment variables [469](#)
- create-ssl command [580](#)
- create-virtual-server command [374](#), [580](#)
- cron [107](#)
  - scheduling execution of logadm [113](#)
- crontab, entry format [112](#)
- custom resources
  - about [231](#)
  - attributes [598](#)
  - creating [231](#)
- custom-resource [598](#)

## D

- daemon
  - native SNMP, restarting [171](#)
- data redundancy unit (DRU) [510](#)
- Data Source Enabled setting [525](#)
- data store [237](#)

- Database Vendor setting 523
- DatabaseName attribute 548
- databases
  - administering and monitoring via CLI 216
  - connection validation 252
  - JDBC API 242
  - JNDI names 222
  - naming services 249
  - resource manager 201
  - resource references 223
  - three-tier access model 243
  - two-tier access model 243
- databuf option 543
- DataBufferPoolSize attribute 548
- datadevices option 517
- DataDeviceSize attribute 533, 548
- DataSource 230
- DataSource Classname setting 523
- DataSource object 244
- dbpassword option 518, 532, 536, 537
- dbpasswordfile option 518, 532, 536, 537
- dbssystempassword option 469
- debug mode
  - starting the application server instance 76
- debug option 76
- default handlers
  - subsystem logging 105
- default HTTP listener
  - Administration Server 58
  - HTTP server 371
- default option values 611
- default web module 377
- delete-acl command 580
- delete-authdb command 580
- delete-auth-realm command 580
- delete-custom-resource command 580
- delete-domain command 65, 581
- delete-file-user command 581
- delete-http-listener command 373, 581
- delete-http-qos command 358, 378, 581
- delete-iiop-cluster-endpoint command 581
- delete-iiop-listener command 581
- delete-instance command 83, 581
- delete-javamail-resource command 581
- delete-jdbc-connection-pool command 581
- delete-jdbc-resource command 581
- delete-jmsdest command 301, 581
- delete-jms-resource command 301, 581
- delete-jndi-resource command 581
- delete-jvm-options command 89
- delete-lifecycle-module command 350, 581
- delete-mime command 361, 581
- delete-persistence-resource command 581
- delete-profiler command 581
- delete-ssl command 581
- delete-virtual server command 581
- delete-virtual-server command 380
- deploy command 346, 477, 581
- deploydir command 346, 477, 581
- deploying
  - COSNaming service with 229
  - directory structure 338
  - disabling 344
  - dynamic 343
  - EJB JAR modules 349
  - individual modules 348
  - lifecycle modules 349
  - redeployment 343
  - RMI/IIOP clients 351
  - runtime environments 340
  - setting availability during 477
  - using Administration interface 347
  - using asadmin 346
  - using Sun ONE Studio 348
  - WAR modules 348
  - web application on cluster 450
- deployment descriptors
  - entries 270
  - J2EE standard 335
  - Sun Java System Application Server 336
  - transaction attributes 209
- deployment, hot
  - deploying an application at server runtime, without restart 184
- description element 641
- destinations, for JMS messages, *See* JMS destinations
- details option 542

- DevicePath attribute 549
- devicepath option 517, 519
- devicesize option 518
- directives, configuring logging 118
- directories, additional document 390
- directory property 503
- directory structure, deployment 338
- disable command 581
- disable-timeout-in-minutes attribute 643, 644
- disabling
  - application server instance in cluster 448, 456
  - deployed applications or modules 344
  - web application in cluster 451, 458
- discardmanualchanges 84
- display-license command 50, 581
- distributable web applications 481
- distributed and local transactions 203
- distributed transactions 245
- DnsBucket
  - monitoring attributes 151
  - monitoring HTTP server elements 148
- document directories
  - additional 390
  - primary 368, 390, 665
  - restricting content publication 395
- document footer, setting 400
- document preferences
  - default MIME type, specifying a 397
  - directory indexing 396
  - index filenames 396
  - parsing the accept language header 375
  - server home page 397
  - virtual servers, setting 395
- document root 368, 665
  - setting 390
- documents, accessed in log 119
- domain directory 62
- domain registry, recreating 67
- domains
  - administrative, about
  - administrative, creating and deleting if non-root users 64
  - configuring 62
  - creating 63

- domains.bin 64
- domains.lck 64
- dotted names, asadmin 584
- DTD files, application XML 336
- dynamic deployment 343
- dynamic reconfiguration 420
- dynamic redeployment
  - redeploying an existing application without a server restart 183
- dynamic reloading 344

## E

- EagerSessionThreshold attribute 549
- EagerSessionTimeout attribute 549
- editing httpd.conf 332
- EJB
  - activation 187
  - cache settings, configuring 195
  - entity beans, about 190
  - MDB pool settings, configuring 196
  - message-driven beans 191
  - module attributes 601
  - passivation 187
  - pool settings, configuring 194
  - references 225
  - session beans, about 189
  - settings, configuring 194
  - types 189
- EJB container
  - about 186
  - attributes 589
  - attributes that can be monitored 193
  - availability configuration 196
  - availability in 475, 480
  - configuring the log level 192
  - responsibilities 188
- EJB JAR file 334
- EJB JAR modules
  - deploying 349
- ejb-container 104, 589
- EJBContext 210
- ejb-jar.xml 226

- ejb-jar.xml deployment descriptor [336](#)
- ejb-link element [226](#)
- ejbLoad [210](#)
- ejb-module [601](#)
- ejb-name element [226](#)
  - mapping [236](#)
- EJBOject [188](#)
- ejb-ref-name element [226](#)
- eliminateRedundantEndTransaction property [524](#)
- enable command [582](#)
- enabled attribute [183](#), [643](#), [644](#)
- enabling
  - application server instance in cluster [448](#), [451](#)
  - web application in cluster [451](#)
- Enterprise Java Beans container, *See* EJB container
- Enterprise Java Beans, *See* EJB
- entity beans
  - about [190](#)
  - bean-managed transactions not allowed [213](#)
  - handling data access via JDBC [189](#)
  - transactions [209](#)
- environment classpath
  - ignore [87](#)
- environment commands, asadmin [565](#)
- environment entries [225](#)
- environment variables
  - AS\_ADMIN\_PREFIX [573](#)
  - asadmin [611](#)
  - ASADMIN\_HOST [567](#)
  - ASADMIN\_INSTANCE [567](#)
  - ASADMIN\_PASSWORD [567](#)
  - ASADMIN\_PORT [567](#)
  - ASADMIN\_SECURE [567](#)
  - ASADMIN\_USER [567](#)
- Error directive [314](#)
- error log file [118](#)
- Error qos-error [159](#)
- error responses, customizing [397](#)
- ErrorLogDateFormat [118](#)
- escape characters
  - asadmin [570](#)
  - in connection pool example [524](#)
- event log file, viewing [119](#)
- event variables, traps [163](#)

- Event Viewer, monitoring events (Windows) [122](#)
- EventBufferSize attribute [549](#)
- events, viewing (Windows) [122](#)
- exceptions, rolling back transactions [210](#)
- execution-time-millis [146](#)
- exit status, asadmin [576](#)
- export command [565](#), [582](#)
- external repositories, accessing [233](#)
- external resources
  - about [231](#)
  - creating [233](#)

## F

- factory objects [226](#)
- Fail All Connections setting [523](#)
- fail-all-connections property [260](#)
- failover of object references [478](#)
- failover scenarios
  - HTTP session [615](#)
  - RMI/IIOP [626](#)
  - stateful session bean [623](#)
- fast option [532](#)
- FATAL [102](#)
- file cache [356](#)
- file manipulation, remote [391](#)
- file persistence type [490](#)
- FINE [102](#)
- FINER [102](#)
- FINEST [102](#)
- FlagAsyncEnabled [151](#)
- FlagCacheEnabled [151](#)
- FlagEnabled [152](#)
- FlagProfilingEnabled
  - monitoring HTTP server attributes [148](#)
- FlagVirtualServerOverflow
  - monitoring HTTP server attributes [149](#)
- flat transactions, J2EE [204](#)
- flexanlg [560](#)
  - use and syntax [120](#)
- FractionSystemMemoryUsage

monitoring Process attributes 150  
frequency option 484

## G

get command 582  
    asadmin 572  
    monitoring data 137  
getUserTransaction 212  
Global Transaction Support setting 523  
Guarantee Isolation Level setting 523

## H

ha persistence type 486  
HADB 507  
    adding machines 534  
    adding nodes 534  
    adding nodes without refragmenting 537  
    architecture 509  
    clearing the database 531  
    configuration 514  
    connection pool properties 523  
    connection pool settings 523  
    creating a database 515  
    creating new management clients 512  
    customer support 554  
    data corruption 554  
    database name 519  
    double networks 520  
    environment variables 555  
    expanding 533  
    expanding nodes 533  
    getting device information 542  
    getting resource information 543  
    getting status of 539  
    getting the JDBC URL 522  
    heterogeneous device paths 519  
    history files 553  
    inetd option 520  
    installing 507  
    listing databases 531

machine maintenance 544  
monitoring 539  
nodes 510, 540  
port assignments 515  
referencing for session persistence 467  
refragmenting 536  
removing a database 532  
restarting a node 528  
restarting the database 531  
rsh or ssh 514  
setting attributes 518, 546  
starting a node 526  
starting the database 529  
stopping a node 527  
stopping the database 530  
using for HTTP session persistence 486  
using for stateful session bean session  
    persistence 480  
hadbm addnodes command 534  
hadbm clear command 531  
hadbm clearhistory command 553  
hadbm command 510  
hadbm create command 515  
hadbm delete command 532  
hadbm deviceinfo command 542  
hadbm get command 546  
hadbm list command 531  
hadbm refragment command 536  
hadbm resourceinfo command 543  
hadbm restart command 531  
hadbm restartnode command 528  
hadbm set command 547  
hadbm start command 529  
hadbm startnode command 526  
hadbm status command 539  
hadbm stop command 530  
hadbm stopnode command 527  
hard links, definition 392  
health checker  
    for cluster 446  
health-checker 425  
health-checker element 645  
help  
    Administration interface 47

- asadmin utility 575
  - help command 582
  - high-availability database (HADB) 507
  - HistoryPath attribute 549
  - historypath option 517
  - home page 397
  - home.html 396
  - Hosts 153
  - hosts attribute
    - checking against subject pattern 368
  - hosts option 519, 536
  - hot deployment
    - deploying an application at server runtime, without restart 184
  - htaccess files 392
  - HTML, server-parsed, setting up 402
  - htpasswd utility 560
  - HTTP 312
    - HTTPS routing 422
    - monitoring 127
    - response timeout, configuring 423
    - session failover 422
    - sessions, persistence of 481
  - HTTP listeners 364
    - acceptor threads, specifying number 59
    - Administration Server 58
    - attributes 604
    - creating 371
    - http-listener-1 364, 371
    - settings 58
    - SSL/TLS security settings, activating 59
  - HTTP server attributes
    - monitoring 148
  - HTTP server elements
    - monitoring 147
  - HTTP service
    - attributes 595
  - HTTP/1.1 protocol 312
  - http-listener 373, 604
  - HTTPS
    - routing 422
    - session failover 422
  - http-server
    - monitoring attribute names 143
  - http-server.http-listener 604
  - http-service 85, 595
  - https-routing property 641
- I
- ibm367 399
  - ibm819 399
  - Id 153
    - monitoring ConnectionQueue attributes 149
    - monitoring HTTP server attributes 148
  - idempotent URLs 487
    - requests 426
  - idle-timeout-in-seconds 145, 196
  - IIOP 438
    - about 304
    - server instance 438
  - IIOP listener
    - attributes 592
    - creating 308
    - ports 310
    - SSL/TLS settings 309
  - IIOP service attributes 591
  - iiop-listener 592
  - iiop-service 105, 591
  - IIS
    - web server plug-in, configuring 324, 325
  - index filename 396
  - index.html 396
  - inetd 520
  - inetd option 518
  - inetdsetupdir option 518, 536
  - in-flight transactions 216
  - inflight-tx 144
  - INFO 102
    - default log level 101
  - INIT 176
  - init.conf 84, 315
    - global variable settings at start-up 359
    - termination timeout 77
  - initial naming context 228
  - initialBeansInPool 193



- init-passthrough 316
- inittab 53, 75, 77
  - editing 78
  - restarting servers automatically 78
  - starting the server with 77
- install-license command 50, 582
- InstallPath attribute 550
- installpath option 516
- instance element 642
- instances
  - application server
    - about 72
    - accessing 49
    - adding 82
    - advanced settings 90
    - applying changes 84
    - deleting 82
    - disabling in cluster 448, 456
    - enabling in cluster 448, 451
    - quiescing in cluster 456
    - reconfiguring in running cluster 461
    - removing from cluster 453
    - requirements for adding to cluster 449
    - restarting (Unix) 77
    - restarting manually (Unix) 75
    - specifying listeners in cluster 448
    - specifying quiescing period for 457
    - starting and stopping 73
    - starting manually 79
    - starting with inittab 77
    - status, viewing 85
    - stopping in cluster 453
    - stopping manually (Unix) 76
- interactive asadmin 565
- Interfaces 153
- internal daemon log rotation 108
- InternalLogbufferSize attribute 550
- interval-in-seconds attribute 645
- IP addresses, in HTTP listeners 364
- IP-address-based virtual servers 366
- isFrozen 144
- iso\_646.irv
  - 1991 399
- iso\_8859-1 399
  - 1987 399
- iso-2022-jp 399
- iso646-us 399
- iso-8859-1 399
- iso-ir-100 399
- iso-ir-6 399
- isolation 200
- iwsCpuID 164
- iwsCpuIdleTime 164
- iwsCpuKernelTime 164
- iwsCpuTable 164
- iwsCpuUserTime 164
- iwsInstanceContact 164
- iwsInstanceCount200 (through 404) 165
- iwsInstanceCount2xx - 5xx 164
- iwsInstanceCount3xx 164
- iwsInstanceCount4xx (& 5xx) 164
- iwsInstanceCount503 167
- iwsInstanceCountOther 165
- iwsInstanceDeathCount 164
- iwsInstanceDescription 164
- iwsInstanceId 164
- iwsInstanceInOctets 164
- iwsInstanceLoad15MinuteAverage 165
- iwsInstanceLoad1MinuteAverage 165
- iwsInstanceLoad5MinuteAverage 165
- iwsInstanceLocation 164
- iwsInstanceNetworkInOctets 165
- iwsInstanceNetworkOutOctets 165
- iwsInstanceOrganization 164
- iwsInstanceOutOctets 164
- iwsInstanceRequests 164
- iwsInstanceStatus 164
- iwsInstanceTable 164
- iwsInstanceUptime 164
- iwsInstanceVersion 164
- iwsListenAddress 167
- iwsListenId 167
- iwsListenPort 167
- iwsListenSecurity 167
- iwsListenTable 167
- iwsProcessConnectionQueueCount 166
- iwsProcessConnectionQueueMax 166

- iwsProcessConnectionQueueOverflows [166](#)
- iwsProcessConnectionQueuePeak [166](#)
- iwsProcessConnectionQueueTotal [166](#)
- iwsProcessId [166](#)
- iwsProcessKeepaliveCount [166](#)
- iwsProcessKeepaliveMax [166](#), [167](#)
- iwsProcessTable [166](#)
- iwsProcessThreadCount [166](#)
- iwsProcessThreadIdle [166](#)
- iwsThreadPoolTable [167](#)
- iwsVsCount200 (through 404) [166](#)
- iwsVsCount2xx - 5xx [165](#)
- iwsVsCount503 [167](#)
- iwsVsCountOther [166](#)
- iwsVsId [165](#)
- iwsVsInOctets [165](#)
- iwsVsOutOctets [165](#)
- iwsVsRequests [165](#)
- iwsVsTable [165](#)

## J

### J2EE

- transactions [200](#)
- Web Container, about [179](#)

### J2EE applications

- EJB specification [279](#)
- JMS, and [279](#)
- message-driven beans, *See* MDB
- resources [219](#)
- services [219](#)
- transactional [203](#)

### J2EE Connectors

- resource manager [202](#)

### J2EE modules

- definition [334](#)
- dynamic reloading [344](#)
- naming [337](#)
- runtime environment [340](#)

### Java Database Connectivity (JDBC) API

- for data access via entity beans [189](#)

### Java Message Service, *See* JMS

### Java Virtual Machine, *See* JVM

- java.io.Serializable interface [478](#)

- java.sql.Connection [212](#)

- java.util.Properties [188](#)

- java-config [590](#)

### JavaMail

- Folder objects [265](#)

- JAF [266](#)

- Message class [265](#)

- Message subclass [266](#)

- Session class [266](#)

- Store class [266](#)

### JavaMail API

- about [262](#)

- message handling [263](#)

### JavaMail resources

- about [262](#)

- attributes [599](#)

- configuration parameters [267](#)

### JavaMail Sessions

- configuring [271](#)

- creating [270](#)

- deployment descriptors [269](#)

- resource factory [226](#)

- javax.ejb.EJBContext [212](#)

- javax.ejb.EntityBean [187](#)

- javax.ejb.EntityContext [187](#)

- javax.ejb.MDBContext [187](#)

- javax.ejb.SessionBean [187](#)

- javax.ejb.SessionContext [187](#)

- javax.ejb.SessionSynchronization [187](#)

- javax.sql.DataSource [203](#)

- javax.sql.XADataSource [203](#)

### JDBC

- API [189](#), [220](#), [242](#)

- connection factories [226](#)

- connections [247](#)

- DataSource object [220](#)

- datasources [220](#), [244](#)

- transactions [261](#)

- URLs [248](#)

### JDBC connection pools

- attributes [597](#)

- connection validation [252](#), [259](#)

- creating [250](#)

- fail-all-connections property 260
- monitoring 260
- pool settings 252
- properties 251
- transaction isolation 252
- JDBC resources
  - attributes 595
  - creating 246
  - registering 246
- jdbc-connection-pool 257, 597
  - monitoring attribute names 144
- jdbc-resource 595
- JdbcUrl attribute 550
- JMS
  - about 274
  - administered objects, *See* JMS administered objects
  - API, list of specifications 191
  - destinations, *See* JMS destinations
  - message consumers 278
  - message delivery models 276
  - message listeners 280
  - message producers 278
  - message structure 276
  - message-driven beans 191
  - messaging system concepts 274
  - physical destinations, *See* JMS destinations
  - programming model 277
  - provider, *See* JMS provider
  - resources, *See* JMS administered objects
  - Service, *See* JMS Service
  - specification 274, 276
  - system architecture 275
- JMS administered objects
  - about 278
  - attributes 598
  - connection factory 285
  - destination 286
  - managing 294
- JMS destinations 221
  - about 284
  - managing 293
  - queues 284
  - topics 284
- JMS provider
  - about 273, 281
  - native 273, 289
  - resource manager 202
- JMS Service
  - administration of 290
  - administration tools 288
  - architecture 287
  - attributes 587
  - built-in 288
  - configuring 291
  - disabling 288
  - external 288
  - MQ administered objects, and 288
  - MQ client runtime, and 288
  - MQ Message Server, and 288
- jms-max-messages-load 145
- jms-ping command 301, 582
- jms-resource 301, 598
- jms-service 105, 301, 587
- JNDI
  - architecture 221
  - connection factories 230
  - custom resources, creating 231
  - external repositories 233
  - external resources, creating 233
  - JMS administered objects, and 285
  - lookup method 221
  - lookup names 338
  - lookups 278
  - lookups and associated references 224
  - MDB and 281
  - name, specifying for session persistence configuration 486
  - names 222
  - resource attributes 596
- JNDI Name setting 525
- jndi-resource 596
- JVM
  - attributes 590
  - debug options 76
  - options 87
  - settings
    - configuring 86, 88
- JVM Profiler
  - attributes 609
  - configuring via Administration interface 88

## K

KeepaliveBucket  
     monitoring HTTP server elements [148](#)  
 keepmanualchanges [84](#)

## L

latin1 [399](#)  
 LDAP, configuring administration server to use [130](#)  
 library, shared, using [352](#)  
 license commands [50](#)  
 lifecycle modules  
     attributes [608](#)  
     deploying [349](#)  
 lifecycle-module [608](#)  
 list command [582](#)  
     monitoring [136](#)  
 list-acls command [582](#)  
 list-authdfs command [582](#)  
 list-auth-realms command [582](#)  
 list-components command [582](#)  
 list-custom-resources command [582](#)  
 list-domains command [65](#), [582](#)  
 listen sockets, *See* HTTP listeners  
 listener  
     HTTP, editing [58](#)  
     specifying for application server instances in  
         cluster [448](#)  
 listeners attribute [643](#)  
 list-file-groups command [582](#)  
 list-file-users command [582](#)  
 list-http-listeners command [372](#), [582](#)  
 list-iiop-cluster-config command [582](#)  
 list-iiop-listeners command [582](#)  
 list-instances command [582](#)  
 list-javamail-resources command [582](#)  
 list-jdbc-connection-pools command [256](#), [582](#)  
 list-jdbc-resources command [258](#), [582](#)  
 list-jmsdest command [301](#), [582](#)  
 list-jms-resources command [301](#), [582](#)

list-jndi-resources command [582](#)  
 list-lifecycle-modules command [350](#), [582](#)  
 list-mimes command [361](#), [582](#)  
 list-persistence-resources command [583](#)  
 list-profilers command [583](#)  
 list-sub-components command [583](#)  
 list-virtual-servers command [583](#)  
 load balancer  
     using multiple [461](#)  
 load balancing  
     described [407](#), [435](#)  
 Load15MinuteAverage  
     monitoring HTTP server attributes [149](#)  
 Load5MinuteAverage  
     monitoring HTTP server attributes [149](#)  
 loadbalancer element [640](#)  
 loadbalancer.xml file [473](#)  
     elements in [639](#)  
     location [637](#)  
 LoadMinuteAverage  
     monitoring HTTP server attributes [149](#)  
 local and distributed transactions [203](#)  
 local option [568](#)  
 local transaction optimization [203](#)  
 location [182](#)  
 locks option [543](#)  
 log analyzer  
     archiving server logs prior to use [120](#)  
     flexanlg, use and syntax [120](#)  
     running [120](#)  
     running from command line [120](#)  
 log archive file format [107](#)  
 log files  
     access [118](#)  
     archiving [106](#)  
     configuring [119](#)  
     error [118](#)  
     virtual servers [370](#)  
 log levels  
     about [101](#)  
     ALERT [102](#)  
     configuring, EJB container [192](#)  
     order of severity [102](#)  
     table of [102](#)

- used for syslog configuration 103
- log rotation
  - internal daemon 107
  - performing (four methods) 107
  - scheduler 108
- log service attributes 593
  - file 116
  - level 116
  - log-stderr 116
  - log-stdout 116
  - use-system-logs 116
- log service element 104
- LOG\_ALERT 103
- LOG\_CRIT 103
- LOG\_DEBUG 103
- LOG\_ERR 103
- LOG\_INFO 103
- LOG\_WARNING 103
- logadm 109
- logadm.conf
  - location and sample of 109
- logbuf option 543
- LogbufferSize attribute 550
- LogFlushInterval 118
- logging
  - about 94
  - access file, viewing 118
  - Application Client Container (ACC) 105
  - client side 105
  - components and subsystems, configuring 117
  - components and subsystems, list of 117
  - configuring attributes via Administration interface 115
  - configuring via administrative interface 114
  - configuring via command line interface 113
  - directives, configuring 118
  - event file, viewing 119
  - features and functions 93
  - messages, information provided 94
  - preferences 119
  - redirecting application and server log output 106
  - UNIX 95
  - using syslog 96
  - virtual server instances 103
  - web container, default behavior 184

- Windows 95
- logical node numbers 539
- log-service 105, 593
- log-virtual-server-id 104
- long options 611

## M

- mail-resource 599
- managed objects 163, 167
- management information base (MIB)
  - defines managed objects 163
- ManagementProtocol attribute 550
- manager-properties element 483, 502
- master agent
  - CONFIG file, editing 174
  - SNMP 162
  - SNMP, enabling and starting 173
  - SNMP, installing 170, 172, 173
  - SNMP, manually configuring 174
  - SNMP, starting 176
  - SNMP, starting on another port 174
  - starting on a nonstandard port 177
- maxBeansInCache 194
- max-beans-in-cache 146
- maxBeansInPool 193
- MaxByteTransmissionRate 154
- MaxCacheEntries 151
- MaxConnections 152
- MaxEntries 152
- MaxHeapCacheSize 152
- MaxMmapCacheSize 153
- MaxOpenConnections 154
- MaxOpenEntries 152
- max-pool-size 145
- MaxProcs
  - monitoring HTTP server attributes 148
- MaxQueued 151
  - monitoring ConnectionQueueBucket attributes 150
- maxSessions property 503

- maxStatement property [524](#)
- MaxTables attribute [550](#)
- MaxThreads [151](#)
  - monitoring HTTP server attributes [148](#)
- MaxVirtualServers
  - monitoring HTTP server attributes [148](#)
- MDB
  - about [191](#), [280](#)
  - deployment descriptor [281](#)
  - JNDI and [281](#)
  - transactions [209](#), [213](#)
- MDB container
  - about [280](#)
  - attributes [588](#)
- MDB pool settings
  - configuring for EJBs [196](#)
- mdb-container [105](#), [588](#)
- memory persistence type [488](#)
- message brokers, *See* MQ brokers
- message listeners [278](#), [280](#)
- message-driven beans, *See* MDB
- MessageListener [191](#)
- messages, logged [94](#)
- messaging
  - JMS *See* JMS
- messaging, asynchronous [273](#)
- meta-data connection validation [259](#)
- metric interval
  - time used in traffic calculations [156](#)
- Microsoft Internet Information Services
  - configuring to use web server plug-in [324](#)
- MIME types
  - attributes [605](#)
  - charset parameter [398](#)
  - creating new file [360](#)
  - definition [360](#), [676](#)
  - editing definitions [361](#)
  - specifying a default [397](#)
  - virtual server settings, configuring [375](#)
  - with virtual servers [375](#)
- mime, dotted name [605](#)
- minBeansInCache [194](#)
- minBeansInPool [193](#)
- Mode [153](#)
  - monitoring Process attributes [150](#)
- modified-attribute persistence scope [499](#)
- modified-session persistence scope [497](#)
- monitoring [151](#), [152](#), [153](#)
  - about [125](#)
  - additional subsystems and components [127](#)
  - bean-cache attributes [145](#)
  - bean-method attributes [146](#)
  - CacheBucket [148](#)
  - CLI name mapping [138](#)
  - client name mapping, example [138](#)
  - ConnectionQueue [147](#)
  - ConnectionQueue server attributes [149](#)
  - ConnectionQueueBucket [147](#)
  - ConnectionQueueBucket attributes [150](#)
  - ConnectionQueueBucket ConnectionQueue attribute [150](#)
  - ConnectionQueueBucket CountOverFlow attribute [150](#)
  - ConnectionQueueBucket CountQueued attribute [150](#)
  - ConnectionQueueBucket CountTotalConnection attribute [150](#)
  - ConnectionQueueBucket CountTotalQueued attribute [150](#)
  - ConnectionQueueBucket MaxQueued attribute [150](#)
  - ConnectionQueueBucket PeakQueued attribute [150](#)
  - ConnectionQueueBucket TicksTotalQueued attribute [150](#)
  - container subsystems [128](#)
  - DnsBucket [148](#)
  - DnsBucket attributes [151](#)
  - FlagProfilingEnabled [148](#)
  - FlagVirtualServerOverflow [149](#)
  - HTTP [127](#)
  - HTTP server attributes [147](#), [148](#)
  - HTTP server elements [147](#)
  - http-server attributes [143](#)
  - Id [148](#), [149](#)
  - JDBC connection pools [260](#)
  - jdbc-connection-pool attributes [144](#)
  - KeepaliveBucket [148](#)
  - Load15MinuteAverage [149](#)
  - Load5minuteAverage [149](#)
  - LoadMinuteAverage [149](#)

- MaxProcs 148
- MaxThreads 148
- MaxVirtualServers 148
- Node Process attribute 150
- object types 141
- ORB service 129
- orb-connection attributes 144
- orb-thread attributes 144
- Pid Process attribute 150
- process 147
- process attributes 143, 150
- Process CountConfigurations attribute 150
- Process FractionSystemMemoryUsage attribute 150
- Process SizeResident attribute 150
- Process SizeVirtual attribute 150
- Profile 147
- Profile attributes 149
- ProfileBucket 148
- quality of service (QOS) 135
- RateBytesReceived 149
- RateBytesTransmitted 149
- RequestBucket 148
- SecondsRunning 148
- server 147
- SNMP 126
- statistics 126
- Thread 148
- ThreadPool 147
- ThreadPool attributes 149
- ThreadPoolBucket 148
- TicksPerSecond 148
- TimeStarted 148
- TimeStarted Process attribute 150
- transaction service 129
- transaction-service attributes 144
- using asadmin to extract data 135
- using get command 137
- using list command 136
- VersionServer 148
- VirtualServer 148
- virtual-server attributes 143

**MQ**

- about
- administered objects 285
- administration tools 286
- brokers 283

- client runtime 285
- integration with Sun Java System Application Server 286
- message server 283
- messaging system, parts of 282
- purchasing MQ Enterprise Edition 282, 636
- resource manager 202
- running trial version of MQ Enterprise Edition 282, 635
- multimode 564, 583
- multiple server pools
  - configuring 326

## N

- name attribute 642, 643
  - for property element 641
- Name setting 523
- NameTrans 314
- naming
  - COSNaming 229
  - initial context 228
  - J2EE modules 337
  - JNDI and resource reference 223
  - JNDI lookup 338
  - services 249
  - standards 337
- native SNMP daemon, restarting 171
- network management station (NMS) 161
  - about 162
- nice 377
- nilogbuf option 543
- node numbers, logical and physical 539
- nodes option 540
- non-interactive asadmin 565
- no-refragment option 535
- no-repair option 527
- nsfc.conf
  - file cache settings 356
- numBeansCreated 194
- numBeansDestroyed 194
- numBeansInPool 194

- num-beans-in-pool [145](#)
- NumberOfDataDevices attribute [550](#)
- NumberOfLocks attribute [544](#), [550](#)
- NumberOfSessions attribute [550](#)
- num-expired-sessions-removed [146](#)
- num-passivation-errors [146](#)
- num-passivations [146](#)
- num-passivation-success [146](#)
- numThreadsWaiting [194](#)
- num-threads-waiting [145](#)

## O

- obj.conf file [84](#)
  - set up SAFs for using quality of service [157](#)
  - template [367](#)
  - virtual server [367](#)
- object references, failover of [478](#)
- object types, monitoring [141](#)
- ObjectType [314](#)
- ObjectType-class [318](#)
- online help
  - Administration interface, accessing [47](#)
  - asadmin utility [575](#)
- online upgrade
  - using multiple clusters for [460](#)
- onMessage [209](#)
- operands, asadmin [563](#)
- optimization, local transaction [203](#)
- options [562](#)
  - boolean [562](#)
  - default values [611](#)
- optionsfile option [469](#), [470](#)
- ORB
  - attributes [591](#)
  - configuring [306](#)
  - functionality of bundled [305](#)
  - IIO listener configuration [308](#)
  - introduction [304](#)
  - listener attributes [592](#)
  - service, monitoring [129](#)
  - thread pools [307](#)

- orb
  - dotted name [591](#)
- orb-connection
  - monitoring attribute names [144](#)
- orblister [592](#)
- orb-thread-pool
  - monitoring attribute names [144](#)

## P

- package-appclient utility [560](#)
- passivation [187](#)
- password file option [567](#)
- password file, loading on startup [395](#)
- password property [523](#)
- password.conf [73](#)
- PathCheck [314](#)
- PeakQueued [151](#)
  - monitoring ConnectionQueueBucket attributes [150](#)
- performance
  - dynamic reloading [344](#)
  - using quality of service (QOS) [135](#)
- persistence
  - about [237](#)
  - bean-managed [190](#)
  - container-managed [191](#)
  - data store and [237](#)
  - entity beans [238](#)
- persistence frequency [492](#)
  - comparison of options [496](#)
  - time-based [494](#)
  - web-method [493](#)
- persistence manager
  - creating [240](#)
  - factory resource attributes [599](#)
  - role of [238](#)
- persistence scope [496](#)
  - comparison of options [501](#)
  - modified-attribute [499](#)
  - modified-session [497](#)
  - session [498](#)
- persistence type



- comparison 492
- file 490
- ha 486
- memory 488
- persistence-manager-factory-resource 599
- physical node numbers 539
- Pid
  - monitoring Process attributes 150
- PidLog 118
- pipe, with asadmin 570
- pkgadd 50
- plug-in, web server, *See* web server plug-in
- Pool Name setting 525
- pool settings, configuring EJB 194
- PooledConnection object 245
- pool-resize-quantity 145
- pools, multiple server
  - configuring 326
- Portbase attribute 550
- portbase option 517
- ports
  - HTTP listener 365
  - IIOP listener 310
- PR\_Recv()/net\_read 160
- PR\_Send()/net\_write 160
- PR\_TransmitFile 160
- preferences, setting log 119
- primary document directory, setting 368, 390, 665
- process
  - attribute 147
  - monitoring attribute names 143
  - monitoring attributes 150
  - monitoring HTTP server elements 147
- Process element 147
- Profile 155
  - monitoring attributes 149
  - monitoring HTTP server elements 147
- Profile element 147
- ProfileBucket
  - monitoring HTTP server elements 148
- ProfileBucket element 147
- profiler 89
  - attributes 609

- dotted name 609
- programming, JMS programming model 277
- properties, about 640
- property element 640
- property option 485
- protocol data units (PDUs) 167
- proxy agent, SNMP 170
  - installing 170
  - starting 171
- public directories
  - configuring 393

## Q

- qos-error, Error 159
- qos-handler, AuthTrans 159
- quality of service 156
  - concurrent connections, virtual servers 161
  - configuring 157
  - configuring for HTTP server 357
  - example 156
  - monitoring 135
  - only HTTP bandwidth for application level
    - measured 159
  - set up SAFs in obj.conf for using 157
  - using 135
  - virtual servers, configuring settings for 377
- queues, *See* JMS destinations
- quiescing 456
  - application 424
  - application server instance in cluster 456
  - configuring 423
  - instance 424
  - web application in cluster 458
- quiescing period 456
  - modifying while quiescing is on 459
  - specifying 448
  - specifying for application server instances 457
  - specifying for web application 451

## R

- ra.xml deployment descriptor [336](#)
- RAR files [334](#)
- RateBytesReceived
  - monitoring HTTP server attributes [149](#)
- RateBytesTransmitted [154](#)
  - monitoring HTTP server attributes [149](#)
- rc.2.d, starting the server with [77](#)
- reapIntervalSeconds property [483](#), [494](#), [503](#)
- recompute interval [156](#)
- reconfig command [68](#), [84](#), [256](#), [573](#), [583](#)
- Recovery, Transaction [204](#)
- redeploying applications [343](#)
- references, failover of [478](#)
- registry, recreating for domains [67](#)
- RelalgdeviceSize attribute [550](#)
- reloading, dynamic [344](#)
- reload-poll-interval-in-seconds property [641](#)
- remote file manipulation, enabling [391](#)
- RemoteException [208](#)
- removal-timeout-in-seconds [195](#)
- request processing, for virtual servers [368](#)
- RequestBucket
  - monitoring HTTP server elements [148](#)
- requests
  - how server handles [312](#)
  - methods [313](#)
  - steps in handling [314](#)
- requirement rules [638](#)
- require-monitor-data property [641](#)
- resource adapter [201](#)
- resource environment references [227](#), [235](#)
- resource manager
  - database [201](#)
  - definition [201](#)
  - J2EE Connectors [202](#)
  - JMS provider [202](#)
  - transaction [202](#)
- Resource RAR files [334](#)
- resource references [223](#), [234](#)
- resources
  - custom [231](#)
  - external [231](#)

- JMS, *See* JMS administered objects
- response timeout, configuring [423](#)
- response-timeout-in-seconds property [641](#)
- res-sharing-scope [205](#)
- restart-instance command [80](#), [583](#)
- restartserv [80](#)
- restricting symbolic links [392](#)
- RMI, about [305](#)
- RMI/IIOP clients, deploying [351](#)
- rollbacks, *See* transactions, rollbacks
- rpm [40](#)
- rsh management protocol [514](#)
- runtime environments [340](#)

## S

- SAF
  - auth-passthrough [316](#)
  - check-passthrough [318](#)
  - init-passthrough [316](#)
  - service-passthrough [317](#)
- sagt, command for starting Proxy SNMP agent [171](#)
- saveto option [553](#)
- scheduler log rotation
  - archive log files [109](#)
  - scheduler link [108](#)
- schedulerd [109](#)
- scope option [484](#)
- script, asadmin [570](#)
- SecondsMaxAge [152](#)
- SecondsRunning
  - monitoring HTTP server attributes [148](#)
- SecondsTimeouts [152](#)
- security audit output, viewing [133](#)
- security service attributes [594](#)
- security, asadmin [578](#)
- security-service [105](#), [594](#)
- Server element [147](#)
- server log output, redirecting [106](#)
- server logs [120](#)
- server.log [94](#)

- default log level 101
- default logging 95
- example 95
- server.xml 84, 99, 104, 106, 214, 343, 355, 364
  - default web application 183
  - enabling availability in 476
  - saving stateful session bean state 480
  - setting the session timeout 482
  - settings that do not require a restart 85
- server1 72
- serverList property 524
- server-parsed HTML 402
- servers configuration attributes 610
- Service 314
- service-passthrough 315, 316, 317
- Services Control Panel
  - using to start the Administration Server 54
- session beans
  - about 189
  - instance variables, synchronizing 211
  - stateful 190
  - stateless 190
  - synchronizing instance variables 211
  - transactions 209, 211
- session persistence 463
  - and single sign-on 481
  - basic configuration steps 464
  - configuring other properties 501
  - default settings 486
  - failover of object references 478
  - for a cluster 485
  - for stateful session beans 479
  - for web applications 481
  - frequency setting 492
  - HTTP, configuring 483
  - other properties for 502
  - referencing HADB database 467
  - scope setting 496
  - specifying cluster name 472
  - specifying JNDI name 486
  - specifying options in file 469
  - to a file 490
  - types 485
  - types, comparison of 492
- session store
  - clearing 470
  - creating 468
  - for stateful session beans 480
  - for web applications 481
  - single or multiple clusters 472
- session timeout 482
- session, persistence scope 498
- sessionFilename property 488, 503
- session-manager element 502
- session-properties element 482, 502
- sessions
  - and dynamic reloading 343, 344
  - JMS messaging 277
- SessionSynchronization 211
- SessionTimeout attribute 550
- session-timeout element 482
- set command 469, 502, 572, 583
- set option 518, 519
- setAutoCommit 212
- setRollbackOnly 210
- settings
  - Administration Server, accessing 57
  - Java Virtual Machine (JVM), configuring 86
- SEVERE 102
- shared library, using 352
- short options 611
- show-component-status command 583
- show-instance-status command 85, 583
- showrev 40
- shutdown command 56, 583
- Simple Network Management Protocol (SNMP)
  - introduction 161
- single sign-on
  - and session persistence 481
- single sign-on, about 184
- singlemode 564
- SizeHeapCache 152
- SizeMmapCache 152
- SizeResident
  - monitoring Process attributes 150
- SizeVirtual
  - monitoring Process attributes 150
- SMUX 169
- SNMP

- community string [168](#)
- community strings, configuring [168](#)
- daemon, restarting [171](#)
- GET and SET messages [167](#)
- master agent [162](#)
  - enabling and starting [173](#)
  - installing [170, 172, 173](#)
  - manually configuring [174](#)
  - starting [176](#)
  - starting on another port [174](#)
- monitoring [126](#)
- native daemon, restarting [171](#)
- proxy agent
  - about [170](#)
  - installing [170](#)
  - starting [171](#)
- setting up on a server [169](#)
- Simple Network Management Protocol, introduction [161](#)
- subagent [162](#)
- subagent, enabling [178](#)
- trap [168](#)
- snmpd, command for restarting native SNMP daemon [171](#)
- soft (symbolic) links [392](#)
- spares option [517, 532, 535](#)
- specify
  - log file [117](#)
  - log level [117](#)
  - transaction log location [118](#)
- SQLTraceMode attribute [550](#)
- ssh management protocol [514](#)
- SSL/TLS
  - HTTP listener settings [59](#)
  - IIOP listener settings [309](#)
  - using with virtual servers [369](#)
- start-appserv command [583](#)
- start-domain command [53, 66, 583](#)
- starting and stopping Apache Web Server [332](#)
- starting the server [77](#)
- start-instance command [74, 76, 583](#)
- startlevel option [526, 528](#)
- StartRepairDelay attribute [551](#)
- startserv [75](#)
  - starting the Administration Server [53](#)
- state, virtual server [376](#)
- stateful session beans
  - session persistence of [479](#)
- StatInterval attribute [551](#)
- statistics
  - monitoring [126](#)
  - quality of service bandwidth lost when server reconfigured dynamically [160](#)
  - settings for measuring traffic [156](#)
- status, application server instance [85](#)
- stderr [95, 106](#)
- stdout [95, 106](#)
- Steady Pool Size setting [523](#)
- steady-pool-size [145](#)
- stop-appserv command [56, 583](#)
- stop-domain command [56, 66, 583](#)
- stop-instance command [74, 583](#)
- stopping the server [56](#)
- stopserv [75](#)
  - shutting down the Administration Server [55](#)
- store option [484](#)
- storepassword option [469](#)
- store-pool-jndi-name property [467](#)
- store-properties element [502](#)
- storeurl option [468](#)
- storeuser option [468](#)
- stronger ciphers [403](#)
- subagent
  - SNMP [162](#)
  - SNMP, enabling [178](#)
- subelements, about [638](#)
- subsystem
  - logging control, at the [104](#)
  - logging default handlers [105](#)
- summary
  - monitorable attribute [143](#)
- Sun customer support [39](#)
- Sun Java System Message Queue, *See* MQ
- Sun Java System Web Server, configuring [319](#)
- Sun ONE Studio [49](#)
  - deploying with [348](#)
- sun-acc.xml [105](#)
- sun-application.xml deployment descriptor [337](#)

- sun-application-client.xml deployment descriptor 337
- sun-cmp-mapping.xml deployment descriptor 337
- sun-ejb-jar.xml deployment descriptor 337
- sun-loadbalancer\_1\_0.dtd file 637
- sun-passthrough.properties 326
  - sample file 327
- sun-web.xml deployment descriptor 337
- sun-web.xml file 182
  - overriding session persistence properties 504
- symbolic (soft) links, definition 392
- symbolic links, restricting 392
- syntax, asadmin 562
- sysContact 174, 175
- sysLocation 174, 175
- syslog
  - info used to identify Application Server messages 99
  - log levels used for configuration 103
  - logging 96
  - message example 100
- syslog.conf 97
  - configuring to store less severe messages 97
  - example of configured file 98
- syslogd 97
- SyslogFacility attribute 551
- SysLogging attribute 552
- SysLogLevel attribute 552
- SyslogPrefix attribute 552
- system RC scripts, used to restart the server automatically 79
- System.getCurrentTimeInMillis 214

## T

- table connection validation 259
- Table Name setting 523
- TakeoverTime attribute 552
- termination timeout
  - init.conf 77
  - setting 77
- Thread
  - monitoring attributes 153
  - monitoring HTTP server elements 148
- thread pools
  - information you specify to add 359
  - ORB 307
- ThreadPool
  - monitoring attributes 149
  - monitoring HTTP server elements 147
- Thread-pool 151
- ThreadPoolBucket
  - monitoring attributes 151
  - monitoring HTTP server elements 148
- thread-pool-size 144
- three-tier database access 243
- TicksDispatch 155
- TicksFunction 155
- TicksPerSecond
  - monitoring HTTP server attributes 148
- TicksTotalQueued
  - monitoring ConnectionQueueBucket attributes 150
- time interval for health check in cluster 446
- time-based persistence frequency 494
- timeout
  - of HTTP sessions 482
  - termination, setting 77
  - value for health check in cluster 446
- timeout-in-seconds attribute 645
- timeoutSeconds property 482, 504
- timeStamp 214
- TimeStarted 153
  - monitoring HTTP server attributes 148
  - monitoring Process attributes 150
- TLS Rollback option
  - ciphers 309
- tools, available for administration functions 44
- topics, See JMS destinations
- total-beans-created 145
- total-beans-destroyed 145
- total-beans-in-cache 146
- total-connections-timed-out 144
- TotalDatadeviceSizePerNode attribute 534, 552
- total-inbound-connections 144

- total-num-calls [146](#)
  - total-num-errors [146](#)
  - total-num-success [146](#)
  - total-outbound-connections [144](#)
  - total-threads-waiting [144](#)
  - total-tx-completed [144](#)
  - total-tx-inflight [144](#)
  - total-tx-rolled-back [144](#)
  - traffic, settings, counting statistics for [156](#)
  - Transaction Isolation setting [523](#)
  - Transaction Manager [200](#)
  - transaction resource manager [202](#)
  - transaction service
    - administering via asadmin [155](#)
    - attributes [587](#)
    - freezing and unfreezing examples [217](#)
    - monitoring [129](#)
  - transactional User Application [201](#)
  - TransactionRequiredException [207](#)
  - transactions
    - administering with Administration interface [214](#)
    - afterBegin method example [211](#)
    - afterCompletion method example [212](#)
    - attributes [206](#), [684](#)
    - bean-managed [193](#)
    - commits [193](#)
    - consistency [200](#)
    - container-managed [205](#)
    - databases, administering and monitoring using
      - asadmin [216](#)
    - distributed [245](#)
    - entity beans [209](#)
    - flat, J2EE [204](#)
    - in-flight [216](#)
    - introduction [199](#)
    - J2EE [200](#)
    - local and distributed [203](#)
    - local optimization [203](#)
    - Mandatory attribute [207](#)
    - message-driven beans [209](#), [213](#)
    - monitoring [217](#)
    - Never attribute [208](#)
    - NotSupported attribute [208](#)
    - recovery [204](#)
    - required attribute [207](#)
    - RequiresNew attribute [207](#)
    - rollbacks [193](#), [210](#), [216](#)
    - session beans [211](#)
    - Supports attribute [208](#)
    - user application [201](#)
  - transactionsCompleted [214](#)
  - transaction-service [105](#), [587](#)
    - monitoring attribute names [144](#)
  - transactionsInFlight [214](#)
  - transactionsRecovered [214](#)
  - transactionsRolledBack [214](#)
  - traps
    - messages containing event variables [163](#)
    - SNMP [168](#)
  - two-tier database access [243](#)
  - type option [484](#)
- ## U
- ulimit [73](#)
  - undeploy command [347](#), [583](#)
  - undeploying a web application from a cluster [452](#)
  - unset command [566](#), [583](#)
  - update-file-user command [583](#)
  - url attribute [645](#)
  - URL connection factory resources [235](#)
  - URL forwarding, configuring [401](#)
  - URL-host-based virtual servers [366](#)
  - URLs, JDBC [248](#)
  - us, alias for us-ascii [399](#)
  - usage, asadmin [578](#)
  - us-ascii [399](#)
  - User Application, transactional [201](#)
  - user directories
    - configuring [393](#)
    - customizing [393](#)
  - user transaction references [228](#)
  - username property [523](#)
  - UserTransaction object [228](#)
  - use-system-logging [97](#)
  - Using Application Server as a proxy [322](#)

using mod\_proxy module 329

## V

Validation Method setting 523

value attribute 641

variables

event traps 163

global, settings in init.conf 359

verifier utility 335, 560

version command 583

VersionServer

monitoring HTTP server attributes 148

virtual servers

acceptor threads 365

attributes 181, 606

concurrent connections, quality of service 161

configuring MIME settings 375

configuring web containers to deploy web applications 181

creating 374

default 367

default configuration example 381

default web application 183

deleting 380

deploying 381

document preferences, setting 395

editing general settings 378

HTTP listeners 364

HTTP listeners, creating 371

intranet hosting example 384

introduction 363

log files 370

logging instances 103

mass hosting example 386

public directories, configuring to use 393

quality of service, configuring settings 377

request processing 368

secure server example 383

setting additional document directories 390

single sign-on 184

state 376

types 366

using access control 370

using quality of service 135

using SSL 369

VirtualServer

monitoring attributes 153

monitoring HTTP server elements 148

virtual-server 379, 606

monitoring attribute names 143

virtual-server attribute 147

VirtualServer element 147

## W

waiting-thread-count 144

WAR files 334, 376

WAR modules, deploying 348

WARNING 102

watchdog 81

web applications 334

deploying on cluster 450

disabling in cluster 451, 458

distributable 481

elements of 181

enabling in cluster 451

quiescing in cluster 458

redeploying on cluster 452

specifying quiescing period for 451

undeploying from cluster 461

with virtual servers 376

web container

attributes 590

availability configuration 185

availability in 475

default logging behavior 184

deploying web applications within virtual servers 181

introduction 179

web application deployment 183

web module attributes 602

web server plug-in 319, 322, 329, 330

about 312

adding 319

configuring 315

- configuring Microsoft Internet Information Services [324](#)
- for Apache [329](#), [332](#)
- IIS, configuring to use [325](#)
- init.conf [315](#)
- web.xml deployment descriptor [336](#)
- web-container [85](#), [104](#), [590](#)
- WEB-INF directory [182](#)
- web-method persistence frequency [493](#)
- web-module [602](#)
  - attributes [182](#)
  - element [643](#)
- wscompile utility [560](#)
- wsdeploy utility [560](#)

## X

- XATransaction mode [202](#)
- x-euc-jp [399](#)
- x-mac-roman [399](#)
- x-sjis [399](#)