

Administrator's Guide

Sun™ ONE Application Server

Version 7, Enterprise Edition

817-1881-10
September 2003

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

Copyright © 2003 Sun Microsystems, Inc. All rights reserved.

THIS SOFTWARE 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, Sun™ ONE, the Java Coffee Cup logo and the Sun™ ONE logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the U.S. and 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.

Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

CE LOGICIEL 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, ÉCRITE ET PRÉALABLE DE SUN MICROSYSTEMS, INC. Droits du gouvernement américain, utilisateurs gouvernementaux - logiciel commercial. Les utilisateurs gouvernementaux sont soumis au contrat de licence standard de Sun Microsystems, Inc., ainsi qu'aux dispositions en vigueur de la FAR (Federal Acquisition Regulations) et des suppléments à celles-ci. L'utilisation est soumise aux termes de la Licence.

Cette distribution peut comprendre des composants développés par des tiers.

Sun, Sun Microsystems, le logo Sun, Java, Sun™ ONE, le logo Java Coffee Cup et le logo Sun™ ONE sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

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.

Contents

About This Guide	21
What's In This Guide?	21
How This Guide Is Organized	22
Part I: Server Basics and Administering Global Settings	22
Part IV: Managing Multiple Server Instances	24
Part V: Appendixes	24
Product Line Overview	25
Platform Edition	25
Standard Edition	26
Enterprise Edition	26
Using the Documentation	26
Documentation Conventions	29
General Conventions	30
Conventions Referring to Directories	31
Product Support	31
 Part 1 Server Basics and Administering Global Settings	 33
Chapter 1 Getting Started with Sun ONE Application Server Administration	35
About Sun ONE Application Server	35
Configuring the Bundled Solaris Version	37
Creating an Administrative Domain	37
Starting the Administration Server	38
Creating an Application Server Instance	38
Deploying Applications	39

Using the Administration Interface	39
Accessing the Administration Interface	39
Using Tabs	42
Using Buttons	42
Accessing Online Help	43
Exiting the Administration Interface	45
Using the Command-line Interface	45
Accessing the Administration Server	45
Accessing Application Server Instances	46
Using Sun ONE Studio	46
About Configuration Files	46
Using the License Commands	47
 Chapter 2 Setting Administration Server Preferences	49
About the Administration Server	50
Starting the Administration Server	51
Using the startserv Script	51
Using the Command-Line Interface	51
Shutting Down the Administration Server	52
Shutting Down Using the Administration Interface	52
Shutting Down Using the stopserv Script	53
Shutting Down Using the Command-Line Interface	53
Accessing the Administration Server Settings	54
Viewing Administration Server Control Settings	56
Applying Changes to the Administration Server	56
Editing HTTP Listener Settings for the Administration Server	57
Setting SNMP, Logging, and Security Preferences	58
 Chapter 3 Configuring Administrative Domains	59
About Administrative Domains	59
Implementing Administrative Domains	60
Directory Structure	60
Process/Port Structure	60
Configuring Domains	60
Creating Domains	61
Example: creating a domain in the default location	61
Example: creating a domain somewhere other than the default location	62
Example: creating a domain for another user (UNIX only)	62
User Permissions on UNIX Platforms	62
Deleting Domains	63
Example: deleting a domain	63
Listing Domains	63

Example: listing the domains on a local machine	64
Example: listing the domains on the local machine using the remote option:	64
Starting Domains	64
Example: starting the only domain on a machine:	64
Stopping Domains	64
Example: Stopping all instances in a domain except for the admin server instance.	65
Recreating the Domain Registry	65

Part 2 Managing an Individual Server Instance 67

Chapter 4 Using Application Server Instances	69
About Application Server Instances	70
Starting and Stopping an Application Server Instance	71
Using the Start and Stop Buttons in the Administration Interface	72
Using the start-instance and stop-instance Commands	72
Using the startserv and stopserv Scripts	72
Starting the Application Server Instance in Debug Mode	73
Setting the Termination Timeout	74
Restarting an Application Server Instance Automatically (UNIX)	74
About Restarting Automatically	75
Restarting Automatically with /etc/inittab (UNIX)	76
Restarting Automatically with the System RC Scripts (UNIX)	76
Restarting an Application Server Instance Manually (UNIX)	76
Restarting the Server Instance Using the Restart Button (UNIX)	77
Restarting the Server Instance Using the restart-instance Command (UNIX)	77
Restarting the Server Instance Using the restartserv Script (UNIX)	78
About the Watchdog	78
Adding an Application Server Instance	79
Deleting an Application Server Instance	80
Applying Changes to an Application Server Instance	81
Viewing Application Server Instance Status	82
Configuring JVM Settings	83
Configuring General Settings	83
Configuring Path Settings	84
Configuring JVM Options	85
Configuring the JVM Profiler	85
Configuring JVM Settings Using the Command-Line Interface	86
Configuring Logging Setting and Monitoring Settings	87
Changing Application Server Instance Advanced Settings	87

Chapter 5 Using Logging	89
About Logging	90
Logging on the UNIX Platform	90
Default Logging in server.log	91
Example of server.log	91
Changing Default Log File Location	92
Logging Using syslog	92
Configuring syslog	93
To configure syslog:	93
Example of syslog messages	96
Using Log Levels	96
About Log Levels	96
Log Levels Used for syslog Configuration	98
About Virtual Servers and Logging	99
About Loggers	101
About Client Side Logging	102
Redirecting Application and Server Log Output	103
Log File Management	103
Internal-daemon Log Rotation	104
Scheduler Based Log Rotation	105
Rotation Using Solaris logadm Utility	106
Rotation Using Solaris “cron” Utility	108
About the crontab Entry Format	109
Using the Solaris cron Utility to Schedule Execution of logadm	110
Configuring Logging Through the Command-line Interface	110
Configuring Logging Through the Administration Interface	111
Configuring the Log Service	112
Configuring Logging for Application Server Components and Subsystems	115
To Specify a Log Level	115
To Specify a Log File: (Virtual Server)	115
To Specify a Transaction Log Location: (Java Transaction Service)	116
Configuring the Directives for Error Logging	116
Viewing the Access Log File	117
Viewing the Event Log File	118
Setting Log Preferences	120
Running the Log Analyzer	121
 Chapter 6 Monitoring Sun ONE Application Server	 125
About Monitoring Sun ONE 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
Quality of Service (QOS)	129
Extracting Monitoring Data Using the CLI	130
The list --monitor Command	130
The get --monitor Command	131
CLI Name Mapping	132
Petstore Example	133
Monitorable Object Types	135
Monitorable Attribute Names	137
HTTP Server Monitorable Objects	142
Monitorable HTTP Server Elements	142
Monitorable HTTP Server Attributes	143
Administering the Transaction Service Using the CLI	151
Using HTTP Quality of Service	152
Quality of Service Example	152
Configuring Quality of Service (QOS)	153
Required Changes to the obj.conf File	156
Known Limitations to Quality of Service	156
About SNMP	158
Network Management Station (NMS)	159
Management Information Base (MIB) Objects	160
SNMP Messages	164
SNMP Trap Destinations	165
SNMP Agent Community	166
Setting Up SNMP	166
Using a Proxy SNMP Agent	167
Installing the Proxy SNMP Agent	168
Starting the Proxy SNMP Agent	169
Restarting the Native SNMP Daemon	169
Installing the SNMP Master Agent	169
Enabling and Starting the SNMP Master Agent	172
Starting the Master Agent on Another Port	172
Manually Configuring the SNMP Master Agent	173
Editing the Master Agent CONFIG File	173
Defining sysContact and sysLocation Variables	174
Configuring the SNMP Subagent	174
Starting the SNMP Master Agent	176
Manually Starting the SNMP Master Agent	176
Starting the SNMP Master Agent Using the Admin Server	176
Enabling the Subagent	178

Chapter 7 Configuring J2EE Containers	181
About the Web Container	181
Understanding the Web Container's Role	183
Web Application Configuration	183
Virtual Server Attributes	184
Web-module Attributes	184
Web Application Deployment	185
Dynamic Re-deployment and Hot Deployment	186
Single Sign-on Facility	186
Logging the Web Container	187
About the EJB Container	188
Understanding the EJB Container's Role	188
Types of Enterprise Java Beans	190
About Message-driven Beans	193
Configuring the EJB Container	194
Performing General Configuration	194
Configuring EJB Settings	196
Configuring MDB Pool Settings	199
 Chapter 8 Using Transaction Services	 201
What Is a Transaction?	201
Transactions in J2EE	202
Transactional Resource Managers	203
Databases	203
JMS Providers	204
J2EE Connectors	204
Local and Distributed Transactions	205
Container-Managed Transactions	207
Transaction Attributes	208
Required	209
RequiresNew	209
Mandatory	209
NotSupported	209
Supports	210
Never	210
Attribute Summary	210
Setting Transaction Attributes	211
Rolling Back a Container-Managed Transaction	211
Synchronizing a Session Bean's Instance Variables	213
Methods Not Allowed in Container-Managed Transactions	214
Bean-Managed Transactions	214
Transaction Service Administration	215
Administering Transactions Using the Administration Interface	215

Administering Transactions Using the Command-Line Interface	218
Listing In-Flight Transactions	218
Managing Transactions	219
Freezing the Transaction Service	219
Monitoring Transactions	219
Chapter 9 Configuring Naming and Resources	221
About J2EE Naming Services and Resources	221
JDBC Datasources	222
Java Mail Sessions	222
JMS Destinations	223
About Java Naming and Directory Interface (JNDI)	223
JNDI Architecture	223
J2EE Naming Services	224
Naming References and Binding Information	225
Naming References in J2EE Standard Deployment Descriptor	226
Application Environment Entries	227
EJB References	227
References to Resource Manager Connection Factories	228
Resource Environment References	229
UserTransaction References	230
Initial Naming Context	230
COSNaming Service	230
JNDI Connection Factories	232
To Create a Custom Resource	233
To Create an External JNDI Resource	235
Accessing External JNDI Repositories	237
Mapping Application Resource References	237
About URL Connection Factory Resources	238
Mapping Application Resource Environment References	238
Mapping EJB References	239
About Persistence Manager Resources	239
What is Persistence?	240
The Role of the Persistence Manager	241
Pre-Deployment Bean Configuration	241
Creating a New Persistence Manager	243
About JDBC Resources	245
About the JDBC API	246
What Does The JDBC API Do?	246
About Database Access Models	247
About JDBC Datasources	248
Properties Of a DataSource Object	248
Registering a JDBC Resource	250

About JDBC Connections	252
About JDBC URLs	253
Configuring JDBC Connection Pools	254
About Connection Pooling	263
Monitoring JDBC Connection Pooling	265
About Connection Sharing	265
About JDBC Transactions	265
About Java Mail Resources	267
About the JavaMail Message-handling Process	268
About the Architectural Components of JavaMail	269
The Message Class	269
Message Storage and Retrieval	270
Message Composition and Transport	270
About JavaBeans Activation Framework (JAF)	271
About JavaMail Configuration Parameters	272
J2EE Deployment Descriptor for JavaMail Session References	274
Entries in Sun ONE Application Server 7, Enterprise Edition Deployment Descriptor	275
Creating a New JavaMail Session	275
Configuring Advanced Resource Properties	277
 Chapter 10 Using the JMS Service	279
About JMS	280
Basic Messaging System Concepts	280
Message	281
Message Service Architecture	281
Message Delivery Models	281
The JMS Specification	282
JMS Message Structure	282
JMS Programming Model	282
Administered Objects: Provider Independence	284
Message-driven Beans	284
The Built-in JMS Service	286
About Sun ONE Message Queue (MQ)	287
Components of an MQ Messaging System	287
MQ Message Server	288
MQ Client Runtime	290
MQ Administered Objects	291
MQ Administration Tools	291
Integration of MQ with Sun ONE Application Server 7, Enterprise Edition	292
Architecture of the Built-in JMS Service	292
Disabling the Built-in JMS Service	294
Administration of the Built-in JMS Service	295
Configuring the JMS Service	296

Managing Physical Destinations	299
Create a Queue or Topic Destination	300
List Physical Destinations	301
Delete a Physical Destination	301
Managing Administered Object Resources	302
Administered Object Attributes	303
Administered Object Resource Management Tasks	303
Administering the Built-in JMS Service Using the Command-Line Interface	308
 Chapter 11 Configuring the Server For Corba/IOP Clients	311
About Support for CORBA/IOP Clients	311
About Interoperability	312
About the ORB	312
About the RMI/IOP Functionality	313
About the Authentication Process	313
Configuring the ORB	314
To Perform General ORB Configuration	314
To Configure IOP Listener For the ORB	317
 Chapter 12 Deploying Applications	321
About J2EE Modules	322
About J2EE Applications	323
J2EE Standard Descriptors	323
Sun ONE Application Server Descriptors	324
Naming Standards	325
Deployment Directory Structure	326
Runtime Environments	328
Module Runtime Environment	328
Application Runtime Environment	329
About Classloaders	330
Deploying Modules and Applications	330
Deployment Names and Errors	331
The Deployment Life Cycle	331
Dynamic Deployment	331
Disabling a Deployed Application or Module	332
Dynamic Reloading	332
Tools for Deployment	333
The asadmin Utility	333
The Administration Interface	335
Sun ONE Studio	335
Deployment of Module or Application	335
Deploying a WAR Module	336

Deploying an EJB JAR Module	336
Deploying a Lifecycle Module	337
The asadmin Utility	337
The Administration Interface	338
Deploying an RMI/IIOP Client	338
Deploying a J2EE CA Resource Adapter	339
Deploying Static Content	339
Access to Shared Frameworks	339
The Application Deployment Descriptor Files	340

Part 3 Managing HTTP Server Features and Virtual Servers 341

Chapter 13 Configuring HTTP Features	343
About the HTTP Features	343
Configuring the File Cache	344
Tuning Your Server for Performance	344
Configuring HTTP Quality of Service	345
Adding and Using Thread Pools	347
Editing Advanced Settings	348
Configuring MIME Types	349
 Chapter 14 Using Virtual Servers	 351
Virtual Servers Overview	351
HTTP Listeners	352
Virtual Servers	353
Types of Virtual Servers	354
IP-Address-Based Virtual Servers	354
URL-Host-Based Virtual Servers	354
Default Virtual Server	355
The obj.conf File	355
Virtual Server Selection for Request Processing	356
Document Root	356
Using Sun ONE Application Server Features with Virtual Servers	357
Using SSL with Virtual Servers	357
Using Access Log Files and Server Log Files	358
Using Access Control with Virtual Servers	358
Using CGIs with Virtual Servers	359
Creating and Configuring HTTP Listeners	359
Creating an HTTP Listener	359
Editing HTTP Listener Settings	361
Deleting an HTTP Listener	361

Creating and Configuring Virtual Servers	362
Creating a Virtual Server	362
Required Settings	363
Optional General Settings	363
Web Application Settings	365
CGI Settings	365
HTTP Quality of Service Settings	365
Editing Virtual Server Settings	366
Editing General Settings Using the Administration Interface	366
Editing General Settings Using the Command-Line Interface	367
Editing CGI Settings	368
Editing Document Handling Settings, Document Directories Settings, and HTTP/HTML Settings	368
Deleting a Virtual Server	368
Deploying Virtual Servers	369
Example 1: Default Configuration	369
Example 2: Secure Server	370
Example 3: Intranet Hosting	372
Example 4: Mass Hosting	374
 Chapter 15 Managing Virtual Server Content	377
Changing the Document Root	378
Setting Additional Document Directories	378
Enabling Remote File Manipulation	379
Using htaccess	380
Restricting Symbolic Links (UNIX)	380
Customizing User Public Information Directories (UNIX)	381
Configuring Public Information Directories	381
Restricting Content Publication	383
Loading the Entire Password File on Startup	383
Setting the Document Preferences	383
Entering an Index Filename	384
Selecting Directory Indexing	384
Specifying a Server Home Page	385
Specifying a Default MIME Type	385
Customizing Error Responses	385
Changing the International Character Set	386
Setting the Document Footer	388
Configuring URL Forwarding	388
Setting up Server-Parsed HTML	389
Setting Cache Control Directives	390
Using Stronger Ciphers	391

Part 4 Configuring Multiple Server Instances 393

Chapter 16 Configuring Load Balancing	395
About Load Balancing	395
Load Balancing Algorithm	396
About Sticky Round Robin Load Balancing Algorithm	396
Requirements for Load Balancing	397
Configuring Load Balancer Plug-in	397
Configuration Changes to Web Server	397
Modifications to Sun ONE Web Server Configuration	398
Modifications to Apache Web Server	399
Creating the Load Balancer Configuration File	401
Configuring Multiple Web Server Instances	403
Configuring Multiple Sun ONE Web Server Instances	404
Configuring Multiple Apache Web Server Instances	405
Configuring Response Timeout	405
Configuring HTTPS Routing	406
About HTTPS Routing	406
Configuring HTTPS Routing	406
Configuring the Health-Checker	407
Configuring Quiescence	408
Quiescing an Instance	408
Quiescing an Application	409
About Dynamic Reconfiguration	409
Monitoring the Load Balancer Plug-in	410
Configuring Log Messages	410
Configuring Monitoring	411
Understanding Monitoring Messages	411
Load Balancer Configuration File	413
The sun-loadbalancer_1_0.dtd File	413
Subelements	414
Data	414
Attributes	415
Elements in the loadbalancer.xml File	415
Known Issues in Load Balancing Requests	420
Chapter 17 Managing Clusters	423
About Clusters	424
Important Facts About Clusters in Sun ONE Application Server	424
Understanding Assigned Requests and Unassigned Requests	426
Defining a Cluster	426
About Adding Application Server Instances to a Cluster	427

Difference Between Enabling and Starting an Application Server Instance	429
Requirements for Adding an Application Server Instance to a Cluster	429
Adding Application Server Instances to a Cluster	430
Setting up Multiple Clusters	431
Deploying a Web Application on a Cluster	432
Sample Cluster Configuration in a loadbalancer.xml File	433
Starting a Cluster	434
Understanding Quiescing	434
Disabling and Quiescing an Application Server Instance in a Cluster	435
Disabling and Quiescing a Web Application in a Cluster	437
Modifying the Quiescing Period While Quiescing is On	438
Using Multiple Clusters for Online Upgrades Without Loss of Service	439
Reconfiguring an Application Server Instance in a Running Cluster	440
Undeploying a Web Application from a Cluster	440
Redeploying an Existing Web Application on a Cluster	441
Stopping an Application Server Instance in a Cluster	441
Stopping a Cluster	442
Removing an Application Server Instance from a Cluster	442
Removing a Cluster	443
Using Multiple Load Balancers	443
 Chapter 18 Configuring Session Persistence	445
About Session Persistence	446
Understanding Persistence Types	446
Configuring Session Persistence for Application Server Instances and Applications	447
Failover of J2EE Object References Stored in HTTP Sessions	448
Basic Session Persistence Configuration Steps	450
Creating a Session Store	451
About Creating a Session Store	452
Creating Session Stores for Multiple Clusters	452
Using the asadmin create-session-store Command to Create a Session Store	454
Configuring JDBC Parameters for the HADB Database	456
Enabling Availability for an Application Server Instance	456
Availability of Single Sign-on Session State	457
Session Availability for Applications Belonging to a Single Sign-On Group	457
Disabling Availability for an Application Server Instance	459
Configuring Session Persistence Options	459
About the configure-session-persistence Command	460
Using cladmin Command to Configure Session Persistence	461
Setting the Persistence Type	462
Setting the Persistence Type to ha	462
Setting the Persistence Type to memory	464
Setting the Persistence Type to file	466

Comparison of Persistence Type Options	468
Setting the Persistence Frequency	469
Setting the Persistence Frequency to web-method	469
Setting the Persistence Frequency to time-based	470
Comparison of Persistence Frequency Options	472
Setting the Persistence Scope	472
Setting the Persistence Scope to modified-session	473
Setting the Persistence Scope to session	474
Setting the Persistence Scope to modified-attribute	476
Comparison of Persistence Scope Options	478
Configuring Other Session Persistence Properties	479
Specifying JNDI Name of the JDBC Resource for HADB Database	481
Specifying a Cluster Name for Application Server Instance	482
Making an Application Distributable	483
Default Values for Session Persistence	484
Clearing the Session Store	485
Before You Clear the Session Store	485
Using the <code>asadmin clear-session-store</code> Command	485
After You Clear the Session Store	487
 Chapter 19 Using the <code>cladmin</code> Command	489
About the <code>cladmin</code> Command	489
<code>asadmin</code> Commands Supported by the <code>cladmin</code> Command	490
Requirements and Limitations	491
Input Files for the <code>cladmin</code> Command	492
The <code>clinstance.conf</code> File	492
The <code>clpassword.conf</code> File	494
Running the <code>cladmin</code> Command	494
Messages	496
Log file for the <code>cladmin</code> Command	496
Using the <code>cladmin</code> Command While Running Multiple Clusters	497
 Chapter 20 Administering the High-Availability Database	499
About the High-Availability Database	500
HADB Architecture	501
HADB Nodes	502
The <code>hadbm</code> Command	502
Configuring the HADB	504
Setting the Management Protocol	504
Creating a Database	505
Additional Steps for <code>inetsd</code>	509
Setting Up the JDBC Connection Pool	510

Getting the JDBC URL	510
Creating a Connection Pool	511
Connection Pool Example	512
Creating a JDBC Resource	513
Managing the HADB	513
Starting a Node	514
Stopping a Node	515
Restarting a Node	517
Starting the HADB	518
Stopping the HADB	518
Restarting the HADB	519
Listing Databases	520
Clearing the HADB	520
Removing a Database	521
Expanding the HADB	522
Adding Storage Space to Existing Nodes	522
Adding Machines	523
Adding Nodes to the HADB	523
Refragmenting the HADB	525
Adding Nodes Without Refragmenting	526
Monitoring the HADB	527
Getting the Status of the HADB	528
Database Status	528
Node Status	529
Getting Device Information	530
Viewing and Modifying Configuration Attributes	533
Getting the Values of Configuration Attributes	533
Setting the Values of Configuration Attributes	533
Configuration Attributes	534
Clearing and Archiving History Files	540
Recovering from Session Data Corruption	542
Using Sun Customer Support for the HADB	542
Environment Variables	543

Part 5 Appendixes 545

Appendix A Using the Command Line Interface	547
About the Command Line Interface	547
About the asadmin Utility	548
About Ant Tasks	548
About Other Command Line Utilities	548

Using asadmin	549
Understanding the Command Syntax	550
Command	550
Options	550
Boolean Options	551
Operands	551
Syntax Example	551
Using Singlemode and Multimode	551
Singlemode	552
Multimode	552
Multiple Multimode	552
Using Interactive and Non-Interactive Options	553
Using the Environment Commands	553
Using the Password File Option	555
Running asadmin Locally or Remotely	556
Using Command Line Invocations	557
Using asadmin from the Command Line	557
Using asadmin with Input from a File (Script)	558
Using asadmin with Standard Input (Pipe)	558
Using Escape Characters	558
Escape Characters on UNIX in Singlemode	559
Escape Characters on All Platforms in Singlemode	559
Escape Characters on All Platforms in Multimode	560
Using get and set Commands	560
get and set Command Examples	561
Getting and Setting Multiple Values Examples	562
Monitoring Using get and set Commands	562
Using Help	563
Viewing Output and Errors	563
Viewing the Exit Status	563
Viewing Usage	564
Security Considerations	564
Concurrent Access Considerations	565
Command Reference	565
List of Commands	565
List of Dotted Names and Attributes	570
Dotted Names Used in asadmin	570
Service Names	570
Resource Names	571
Application Names	572
Other Names	572
Attributes	573
jms-service	573

transaction-service	574
mdb-container	574
ejb-container	575
web-container	576
java-config	577
orb or iiop-service	577
orblister or iiop-listener	579
log-service	580
security-service	580
http-service	581
jdbc-resource	582
jndi-resource	582
jdbc-connection-pool	583
custom-resource	584
jms-resource	585
persistence-manager-factory-resource	585
mail-resource	586
application	587
ejb-module	587
web-module	589
connector-module	589
http-listener or http-server.http-listener	590
mime	591
acl	592
virtual-server	593
auth-db	594
authrealm	594
lifecycle-module	595
profiler	596
server configuration (name of server instance)	596
Long and Short Option Formats, Default Values, and Environment Variable Equivalents	597
 Appendix B Failover Scenarios	 603
Assumptions and Requirements	603
Persistence Type Set to memory for Application Server Instances in Cluster	604
Persistence Type Set to memory for an Application Server Instance	605
Persistence Type Set to file for an Application Server Instance	606
Persistence Type Set to ha for Application Server Instances in a Cluster	607
Persistence Type Set to memory for a Web Application	608
Persistence Type Set to file for a Web Application	609
Persistence Type Set to ha for a Web Application	609

Appendix C Compiling and Configuring Apache Web Server	611
Minimum Requirements	611
Installing SSL-aware Apache	612
Compiling and Building Open SSL	612
Configuring Apache with mod SSL	612
Compiling and Building Apache	613
Starting and Stopping Apache	614
 Appendix D Running Sun ONE Message Queue Broker with Trial Enterprise License ..	615
About MQ	615
Running MQ Broker with Trial License of Enterprise Edition of MQ	616
Purchasing the Enterprise Edition License of MQ	616
 Appendix E Third Party Copyright Notices	619
 Glossary	621
 Index	651

About This Guide

This guide describes how to configure and administer Sun[™] ONE Application Server 7, Enterprise Edition. 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 includes the following sections:

- [What's In This Guide?](#)
- [How This Guide Is Organized](#)
- [Documentation Conventions](#)
- [Product Line Overview](#)
- [Documentation Conventions](#)
- [Product Support](#)

What's In This Guide?

This guide explains how to configure and administer the Sun ONE Application Server. After configuring your server, use this guide to help maintain your server.

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, [Appendixes](#) addresses specific reference topics that describe various topics, including internationalization issues, server extensions, failover scenarios, and the Sun ONE Application Server command line interface documentation.

Part I: [Server Basics and Administering Global Settings](#)

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

- [Chapter 1, “Getting Started with Sun ONE Application Server Administration,”](#) provides an overview of Sun ONE 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 Application Server Instances,”](#) describes how to configure server preferences for your Sun ONE Application Server.

- [Chapter 5, “Using Logging,”](#) describes the foundation for logging, and the logging features and functions within Sun ONE Application Server.
- [Chapter 6, “Monitoring Sun ONE Application Server,”](#) contains information about the monitoring and Simple Network Management Protocol (SNMP) features and functions available within Sun ONE Application Server 7, Enterprise Edition.
- [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 ONE 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 Sun ONE Application Server 7, Enterprise Edition environment.
- [Chapter 12, “Deploying Applications,”](#) describes how to deploy applications to the Sun ONE 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 13, “Configuring HTTP Features,”](#) describes how to configure preferences for your HTTP-related features of your Sun ONE Application Server.
- [Chapter 14, “Using Virtual Servers,”](#) describes how to set up and administer virtual servers using your Sun ONE Application Server.
- [Chapter 15, “Managing Virtual Server Content,”](#) describes how you can configure and manage your server’s content.

Part IV: Managing 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 16, “Configuring Load Balancing,”](#) describes how to setup load balancing of HTTP requests, configure, and administer the load balancer plug-in.
- [Chapter 17, “Managing Clusters,”](#) 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 Sun ONE Application Server 7, Enterprise Edition.
- [Chapter 18, “Configuring Session Persistence,”](#) describes how to configure session persistence for the Sun ONE Application Server.
- [Chapter 19, “Using the cladmin Command,”](#) describes the use of the `cladmin` command that is bundled with the product.
- [Chapter 20, “Administering the High-Availability Database,”](#) describes the High Availability Database (HADB) in the Sun ONE Application Server 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,”](#) 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,”](#) explains how to compile and build Apache Web Server to work with Sun ONE Application Server 7, Enterprise Edition.
- [Appendix D, “Running Sun ONE Message Queue Broker with Trial Enterprise License,”](#) describes how to run the Sun ONE Message Queue (MQ) broker using a trial enterprise license so that you can evaluate enterprise features of the MQ broker.

- [Appendix E, “Third Party Copyright Notices,”](#) contains additional copyright information.

Product Line Overview

Sun ONE 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 ONE 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. Three editions of the application server are offered to suit a variety of needs for both production and development environments:

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

Platform Edition

Platform Edition forms the core of the Sun ONE Application Server 7 product line. This product offers a high-performance, Java™ 2 Platform, Enterprise Edition (J2EE™ platform) 1.3 specification-compatible runtime environment that is ideally suited for basic operational deployments, as well as for embedding in third-party applications.

Platform Edition deployments are limited to single application server instances (that is, single virtual machines for the Java platform (Java™ virtual machine or JVM™)). Multi-tier deployment topologies are supported by the Platform Edition, but the web server tier proxy does not perform load balancing. In Platform Edition, administrative utilities are limited to local clients only.

The Platform Edition of Sun ONE Application Server 7 is bundled with Solaris 9.

Standard Edition

The Standard Edition builds on the functionality of the Platform Edition, and 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. Sun ONE Directory Server is bundled with Standard Edition for user authentication and limited application configuration storage.

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 ONE 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.

Using the Documentation

The Sun ONE Application Server 7, 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 ONE Application Server manuals. The left column lists the tasks and concepts, and the right column lists the corresponding manuals.

Table 1 Sun ONE Application Server 7, Enterprise Edition Documentation Roadmap

For information about	See the following
Late-breaking information about the software and the documentation.	<i>Release Notes</i>
Comprehensive, table-based summary of supported hardware, operating system, JDK, and JDBC/RDBMS.	<i>Platform Summary</i>
Sun ONE Application Server 7 overview, features available with each product edition.	<i>Product Overview</i>
Diagrams and descriptions of server architecture, benefits of the Sun ONE Application Server architectural approach.	<i>Server Architecture</i>
New enterprise, developer, and operational features of Sun ONE Application Server 7.	<i>What's New</i>
How to get started with the Sun ONE Application Server 7 product. Includes new features, architectural overview, and sample application tutorial.	<i>Getting Started Guide</i>
Installing the Sun ONE Application Server software and its components, such as sample applications, the Administration interface, and the high-availability components. Instructions for implementing a basic high-availability configuration are included.	<i>Installation Guide</i>
Evaluating your system needs and enterprise to ensure that you deploy Sun ONE 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>
Best practices for HTTP session availability that application architects and developers can use.	<i>Application Design Guidelines for Storing Session State</i>
Creating and implementing Java™ 2 Platform, Enterprise Edition (J2EE™ platform) applications intended to run on the Sun ONE Application Server 7 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 ONE Application Server glossary is included.	<i>Developer's Guide</i>

Table 1 Sun ONE Application Server 7, Enterprise Edition Documentation Roadmap

For information about	See the following
Creating and implementing J2EE web applications that follow the Java™ Servlet and JavaServer Pages (JSP) specifications on the Sun ONE Application Server 7. 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>
Creating and implementing J2EE applications that follow the open Java standards model for enterprise beans on the Sun ONE Application Server 7. 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>
Creating Application Client Container (ACC) clients that access J2EE applications on the Sun ONE Application Server 7.	<i>Developer's Guide to Clients</i>
Creating web services in the Sun ONE Application Server environment.	<i>Developer's Guide to Web Services</i>
Java™ Database Connectivity (JDBC™), transaction, Java Naming and Directory Interface™ (JNDI), Java™ Message Service (JMS), and JavaMail™ APIs.	<i>Developer's Guide to J2EE Services and APIs</i>
Creating custom NSAPI plug-ins.	<i>Developer's Guide to NSAPI</i>
Information and instructions on the configuration, management, and deployment of the Sun ONE 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 ONE Application Server glossary is included.	<i>Administrator's Guide</i>
Editing Sun ONE Application Server configuration files, such as the <code>server.xml</code> file.	<i>Administrator's Configuration File Reference</i>

Table 1 Sun ONE Application Server 7, Enterprise Edition Documentation Roadmap

For information about	See the following
Configuring and administering security for the Sun ONE 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 ONE Application Server 7. Topics include the Administration Tool, Pooling Monitor, deploying a JCA connector, and sample connectors and sample applications.	<i>J2EE CA Service Provider Implementation Administrator's Guide</i>
Migrating your applications to the new Sun ONE Application Server 7 programming model, specifically from iPlanet Application Server 6.x and from Netscape Application Server 4.0. Includes a sample migration.	<i>Migrating and Redeploying Server Applications Guide</i>
How and why to tune your Sun ONE Application Server to improve performance.	<i>Performance Tuning Guide</i>
Information on solving Sun ONE Application Server problems.	<i>Troubleshooting Guide</i>
Messages that you may encounter while running Sun ONE Application Server 7. Includes a description of the likely cause and guidelines on how to address the condition that caused the message to be generated.	<i>Error Message Reference</i>
Utility commands available with the Sun ONE Application Server; written in manpage style.	<i>Utility Reference Manual</i>
Using the Sun™ Open Net Environment (Sun ONE) Message Queue software.	The Sun ONE Message Queue documentation at: http://docs.sun.com/db?p=prod/sl.slmsgqu

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).

- **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 31](#)

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

- Solaris[™] 8 non-package-based Evaluation installations:

user's home directory/sun/appserver7

- Solaris unbundled, non-evaluation installations:

/opt/SUNWappserver7

The directories, *default_config_dir* and *install_config_dir* are identical to *install_dir*. See [“Conventions Referring to Directories” on page 31](#), 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*

Conventions Referring to Directories

By default, when using the Solaris™ 8 and 9 installation, the application server files are spread across several root directories. These directories are described in this section.

- **For Solaris 8 and 9 installations**, 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.

Product Support

If you have general feedback on the product or documentation, please send this to appserver-feedback@sun.com.

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
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps

Server Basics and Administering Global Settings

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

Chapter 2, “Setting Administration Server Preferences”

Chapter 3, “Configuring Administrative Domains”

Getting Started with Sun ONE Application Server Administration

This chapter describes the basics of administering Sun ONE 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 ONE Application Server](#)
- [Configuring the Bundled Solaris Version](#)
- [Using the Administration Interface](#)
- [Using the Command-line Interface](#)
- [Accessing the Administration Server](#)
- [Accessing Application Server Instances](#)
- [Using Sun ONE Studio](#)
- [About Configuration Files](#)
- [Using the License Commands](#)

About Sun ONE Application Server

The Sun ONE 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 ONE 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 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 ONE Application Server architecture and features, and for initial steps to take with your Sun ONE Application Server, see the *Sun ONE Application Server Getting Started Guide*.

Configuring the Bundled Solaris Version

This guide refers to two kinds of Sun ONE Application Server installations for Solaris: Solaris 9 bundled and unbundled. If you received your copy of the Sun ONE Application Server as part of the Solaris 9 installation, you have the Solaris bundled version. If you have a standalone copy of the Sun ONE Application Server, you have the unbundled version.

NOTE If you are using the unbundled Solaris version of the Sun ONE Application Server, please skip this section and move on to [“Using the Administration Interface” on page 39](#).

If you are using the version of the Sun ONE Application Server bundled with Solaris 9, you need to perform some extra configuration steps before you can start using the server.

When you install an unbundled version of the Sun ONE Application Server, as part of the installation process a domain, an Administration Server, and a server instance are automatically created.

When you use the Solaris 9 bundled version, you have to create these items manually, as well as perform other steps. Once you have performed these initial steps, you can take advantage of all Sun ONE Application server features, including adding additional administrative domains and server instances.

The following topics are discussed in this section:

- [Creating an Administrative Domain](#)
- [Starting the Administration Server](#)
- [Creating an Application Server Instance](#)
- [Deploying Applications](#)

Creating an Administrative Domain

Multiple administrative domains allow 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. Each domain has one Administration Server.

When you create a new domain, you specify:

- A port number for the Administration Server. The default when you install an unbundled version is 4848.
- An administration username and password. These passwords are required when you access the Administration Server, either when you access the Administration interface or when you run the command-line interface.
- The domain location.

You must create a domain using the command-line interface's `asadmin` utility's `create-domain` command. For more information on creating an administrative domain, see [Chapter 3, “Configuring Administrative Domains.”](#)

Starting the Administration Server

When you create an administrative domain, you create an Administration Server. The Administration Server is a special instance of the Sun ONE Application Server that serves the Administration interface and provides administrative facilities for the command-line interface.

In order to use the Administration interface or to use many of the commands in the command-line interface, you must have a running Administration Server. For information on starting the Administration Server, see [“Starting the Administration Server” on page 51.](#)

Creating an Application Server Instance

Once you have created a domain and started the Administration Server, you need to create an application server instance. Each application server instance has its own J2EE configuration, J2EE resources, application deployment areas, and server configuration settings.

You can create an application server instance through the Administration interface or through the command-line interface. The server instance is created in a folder within the domain.

On the unbundled version, the server instance created at installation is called `server1`. You will often see `server1` used in examples throughout the documentation.

For more information about creating an application server instance, see [“Adding an Application Server Instance” on page 79.](#)

Deploying Applications

Once you have created a domain, started the Administration Server, and added an application server instance, you can deploy applications to that instance. For more information, see [Chapter 12, “Deploying Applications.”](#)

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 ONE Application Server runs on an HTTP server called the Administration Server. The Administration Server is installed when you install Sun ONE Application Server if you are using an unbundled version. You have to create an administrative domain and Administration Server if you are installing the bundled version. For more information, see [“Configuring the Bundled Solaris Version” on page 37.](#)

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 51.](#)

When you installed the Sun ONE 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/`

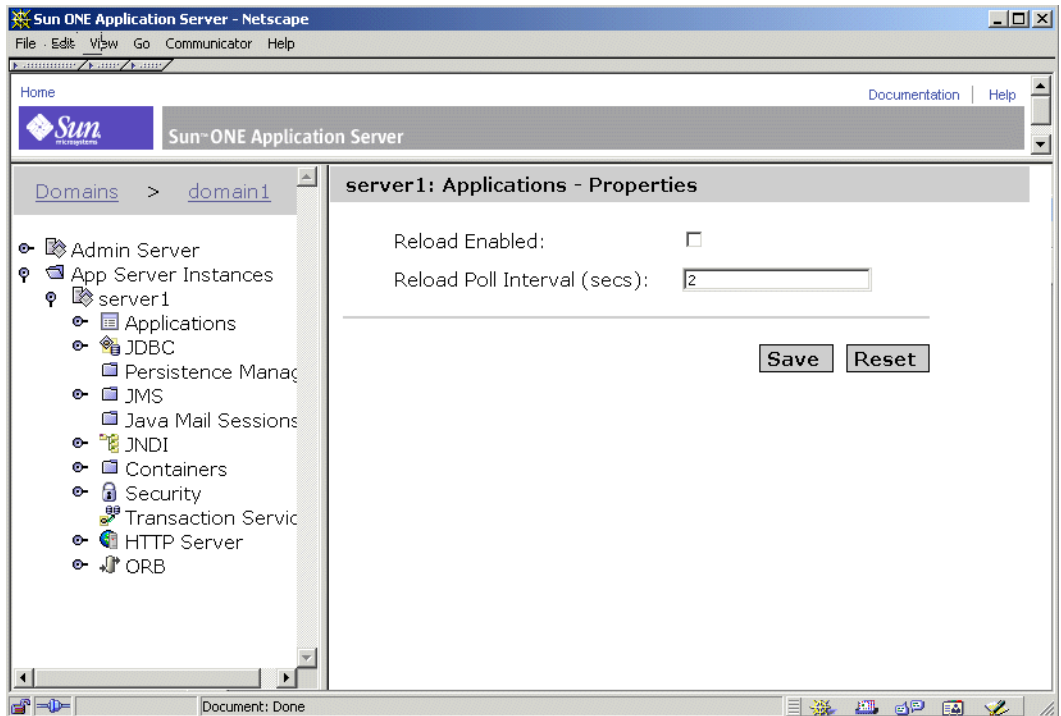
If you are accessing the Administration Server from the same machine you installed Sun ONE Application Server on, you can use:

`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.

The following figure shows the Administration interface.

Figure 1-1 The Sun ONE Application Server Administration Interface

The left pane is a tree view of all items you can configure in the Sun ONE 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 1-2 Closed Symbol

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

Figure 1-3 Open Symbol

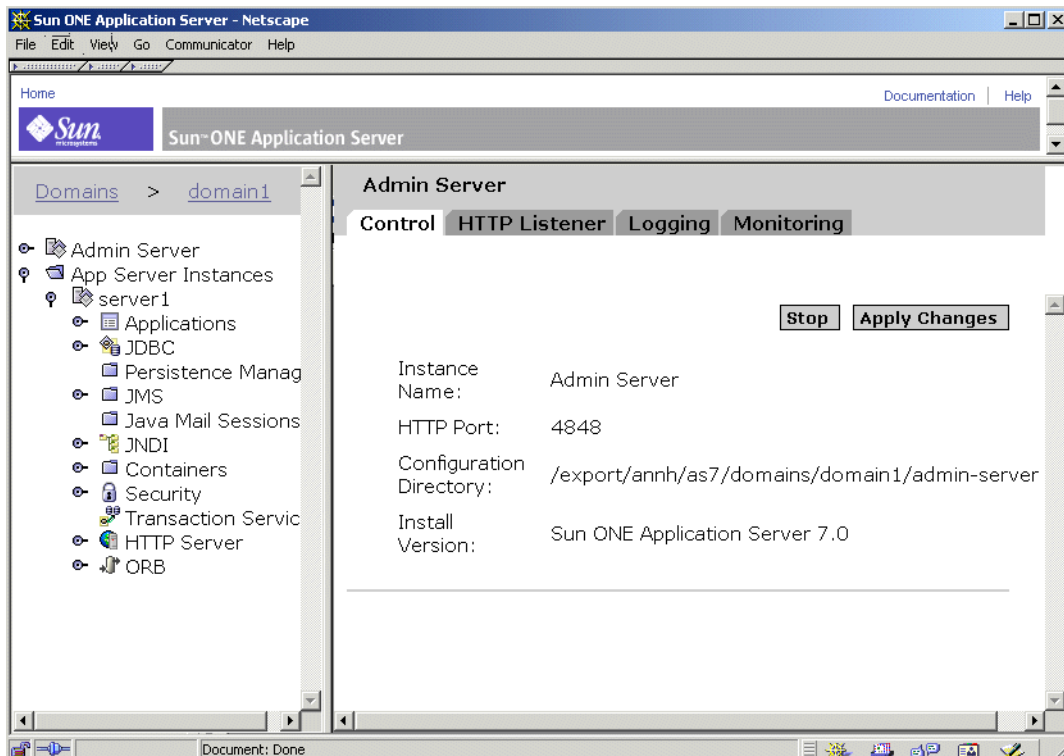
Using Tabs

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.

The following figure shows the Administration interface with tabs:

Figure 1-4 Administration Interface with Tabs



Using Buttons

The following standard buttons are available in the Administration interface.

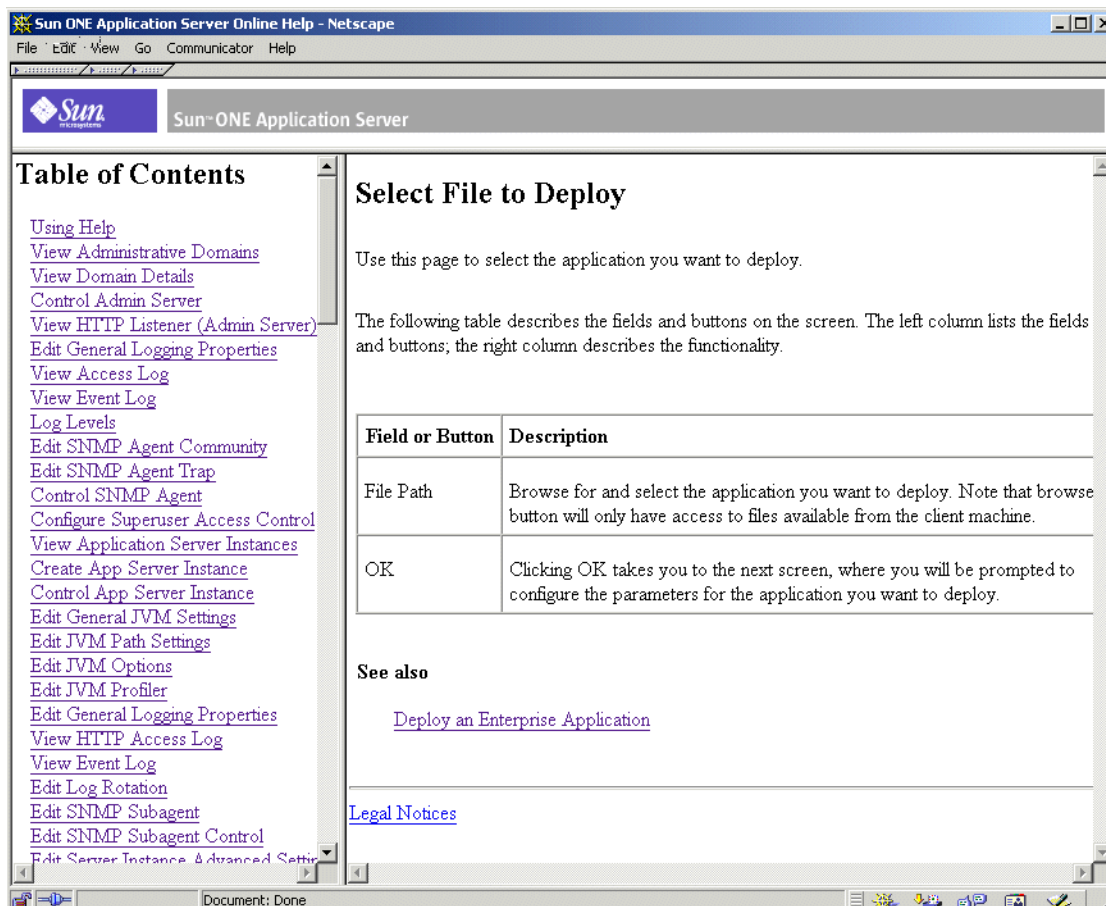
Table 1-1 Standard Administration interface Buttons

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 Application Server Instances page displays the Create New Instance page.
OK	Saves your entries. If you've made configuration changes to a Sun ONE Application Server instance, you must apply your changes in order for them to take effect. For more information, see “Applying Changes to an Application Server Instance” on page 81 .
Reset	Resets the values on the page to the default values.
Save	Saves your entries. If you've made configuration changes to a Sun ONE Application Server instance, you must apply your changes in order for them to take effect. For more information, see “Applying Changes to an Application Server Instance” on page 81 .

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.

Figure 1-5 Online Help

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 ONE 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 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 ONE 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 Application Server Instances

You can have multiple Sun ONE 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 do 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 if you are using the unbundled version of the software. You can create others if you like.

If you are using Solaris 9 bundled version, the application server instance is not automatically created. For more information, see [“Configuring the Bundled Solaris Version” on page 37](#).

For more information on application server instances, see [Chapter 4, “Using 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 4. For more information about Sun ONE Studio, see the *Sun ONE Studio 4, Enterprise Edition for Java with Application Server 7 Tutorial* 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 ONE Application Server Administrator’s Configuration File Reference*.

Using the License Commands

When you purchase Sun ONE 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 ONE Application Server Installation Guide*.

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

For unbundled Solaris package-based installations, the following syntax is used for license installation:

```
pkgadd -d full_path SUNWaslco
```

For example,

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

For Solaris 9 bundled license installation, the following syntax is used for license installation:

```
pkgadd -d full_path SUNWaslc
```

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

The Administration Server is a special instance of the Sun ONE 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 the Sun ONE Application Server. This chapter explains how to configure the Administration Server.

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 ONE 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 ONE Application Server (unbundled version) or create a new domain. You can only have one Administration Server per domain, though you may create multiple Sun ONE Application Server instances that are managed by an Administration Server. You can access configuration settings, deployed applications, and other server features for each application server instance by using the Administration Server.

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

If you are using the unbundled Sun ONE Application Server, when you installed the Sun ONE Application Server you chose a port number for the Administration Server, or used the default port of 4848.

If you are using the Solaris 9 bundled Sun ONE Application Server, you must create a domain and an Administration Server manually. For more information on configuring the Solaris 9 bundled version, see [“Configuring the Bundled Solaris Version” on page 37.](#)

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

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

Note that *domain* here is not your Sun ONE 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 ONE 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:

- [Using the startserv Script](#)
- [Using the Command-Line Interface](#)
- [Shutting Down the Administration Server](#)

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`.

NOTE	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.
-------------	--

Using the Command-Line Interface

The command-line interface's `asadmin` utility has a command `start-domain` that you can use to start your Application Server and all associated Sun ONE Application Server instances. You can only run this command locally, that is, from the machine where your Sun ONE 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.”](#)

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](#)

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.

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 ONE 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 ONE 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:

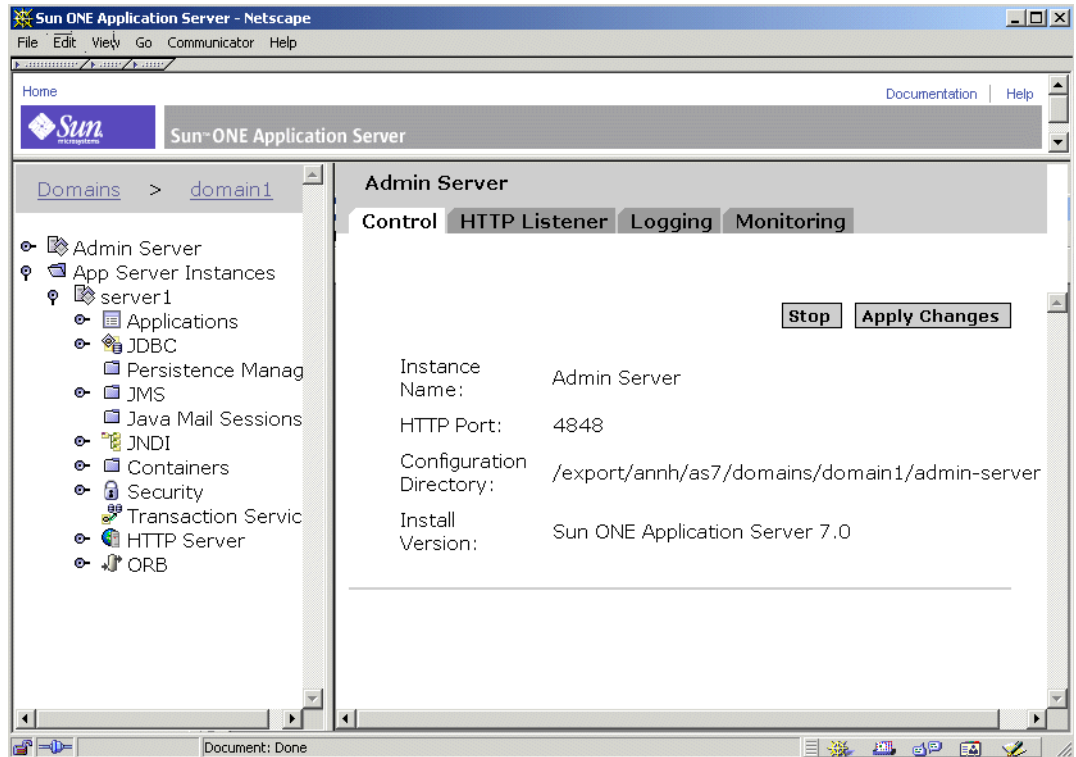
```
asadmin stop-domain [--user admin_user] [--password admin_password]  
[--host admin_host] [--port admin_port] [--local=true/false] [--domain  
domain_name] [--adminserv=true/false] [--passwordfile file_name] [--secure |  
-s]
```

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.”](#)

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.

Figure 2-1 Administration Server User Interface

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 ONE 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 ONE 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 2-2 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.

With the unbundled version of the Sun ONE Application Server, an HTTP listener for the Administration Server, `http-listener-1`, is created automatically when you install. 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.) For more information about configuring the Solaris 9 bundled version, see [“Configuring the Bundled Solaris Version” on page 37](#).

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 ONE Application Server Performance Tuning and Sizing 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 ONE Application Server.”](#) For information on logging, see [Chapter 5, “Using Logging.”](#) For information on security settings, see the *Sun ONE Application Server Administrator’s Guide to Security*.

Configuring Administrative Domains

This chapter explains how to set up and administer administrative domains with your Sun ONE Application Server 7, Enterprise Edition.

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 One 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 don't 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 62](#).

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 a script goes wrong, or 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 <code>asadmin</code> 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're 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 Application Server Instances”

Chapter 5, “Using Logging”

Chapter 6, “Monitoring Sun ONE 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, “Deploying Applications”

Using Application Server Instances

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

This chapter includes the following topics:

- [About 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 Application Server Instances

Sun ONE Application Server creates one application server instance, called `server1`, when you install the unbundled version of the software. You can delete the `server1` instance and create a new instance with a different name if you prefer.

If you are using the Solaris 9 bundled version of the software, you must create your own server instance. For more information, see [“Configuring the Bundled Solaris Version” on page 37](#).

Each 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 ONE 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 14, “Using Virtual Servers.”](#)

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

Starting and Stopping an Application Server Instance

A Sun ONE Application Server instance is not started automatically. Once you start an instance, it 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 startserv and stopserv Scripts](#)

NOTE	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 ONE Application Server prompts the administrator for the key database password before starting up. If you want to be able to restart an unattended Sun ONE 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 ONE Application Server Administrator's Configuration File Reference*.

NOTE	On UNIX, some Sun ONE 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.
3. 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 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.

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

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.

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 71](#) and [“Deleting an Application Server Instance” on page 80](#).

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 78](#).

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 81](#).

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.
3. 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:

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.

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) is a program that is shipped with your Sun ONE 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: 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 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've 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 it 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 ONE 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 ONE 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 you 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 ONE 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 ONE 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 the *Sun ONE Application Server Developer's Guide to Enterprise Java Beans*.

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 ONE 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 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 ONE 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 ONE 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 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 ONE 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 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 application server's own log output can help identify and diagnose server configuration and deployment problems.

Logging within the Sun ONE Application Server uses the Java logging API. Sun ONE 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 112](#).

Logging on the UNIX 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](#)

Default Logging in server.log

On UNIX 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 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 ONE 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 ONE Application
Server 7.0
```

```
[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 ONE Application Server 7, Enterprise Edition instance's `server.log` file, change the path to the log file in the Administrator Interface. See [Configuring the Log Service](#) 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 ONE 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. You can direct logging to `syslog` by enabling this attribute in the Server element of `server.xml` described in the Sun ONE 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. Give 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
.warning/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.erroperator
#*.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.debugifdef(`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 ONE Application Server for the change to take effect.

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

- Unique message ID
- Timestamp
- Instancename
- 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 99](#). The log service configuration for the server instance is described in [“Configuring Logging Through the Administration Interface” on page 111](#).

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 ONE Application Server 7.0

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
```

Using Log Levels

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

The following topics are described:

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

About Log Levels

The Sun ONE Application Server uses the standard JDK 1.4 log levels for selective logging of information. In addition to the standard JDK log levels, Sun ONE 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 ONE Application Server Log Levels Mapped to server.log” on page 99](#).

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 ONE 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 ONE Application Server Administrator's Configuration File Reference*, section “Manually Editing Configuration Files”.

An example of setting the log level through the Administration interface is shown in the figure [“Log Level for JMS Service” on page 102](#). To set the log level for each subsystem or component directly in the `server.xml` file, see the *Sun ONE Application Server Administrator's Configuration File Reference*.

The Log levels described in the table [“Log Levels” on page 98](#) match the requirements of the JDK1.4 logging API specification. However, the log levels ALERT and FATAL are specific to Sun ONE Application Server and are not implemented in the JDK1.4 logging API.

The following table defines the log levels and messages in Sun ONE Application Server, in increasing order of severity. The left column lists the log level designation in the Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server.

NOTE	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 ONE Application Server 7, Enterprise Edition when using `syslog`. The left column lists the log level designation in the Sun ONE Application Server 7, Enterprise Edition, and right column provides the corresponding log level in the `syslog` facility.

Table 5-2 Sun ONE Application Server Log Levels Mapped to server.log

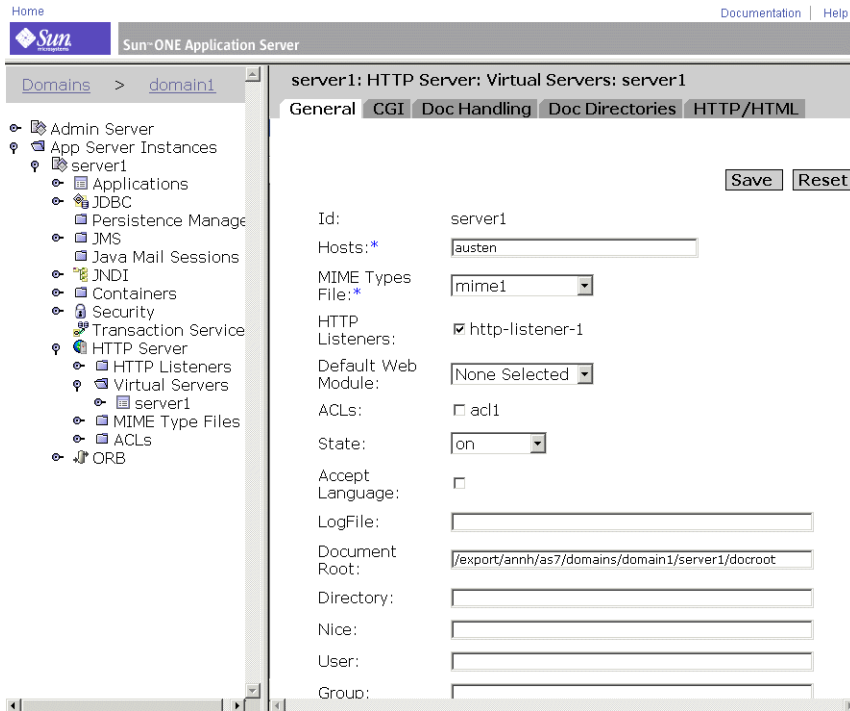
Sun ONE Application Server	syslog level
FINEST	LOG_DEBUG
FINER	LOG_DEBUG
FINE	LOG_DEBUG
CONFIG	LOG_INFO
INFO (default)	LOG_INFO
WARNING	LOG_WARNING
SEVERE	LOG_ERR
ALERT	LOG_ALERT
FATAL	LOG_CRIT

About Virtual Servers and Logging

The Sun ONE Application Server can have virtual server instances. Each virtual server within a Sun ONE 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. The figure [“Setting the Virtual Server Log File Name” on page 100](#) shows where this setting is located.

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

Figure 5-1 Setting the Virtual Server Log File Name

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

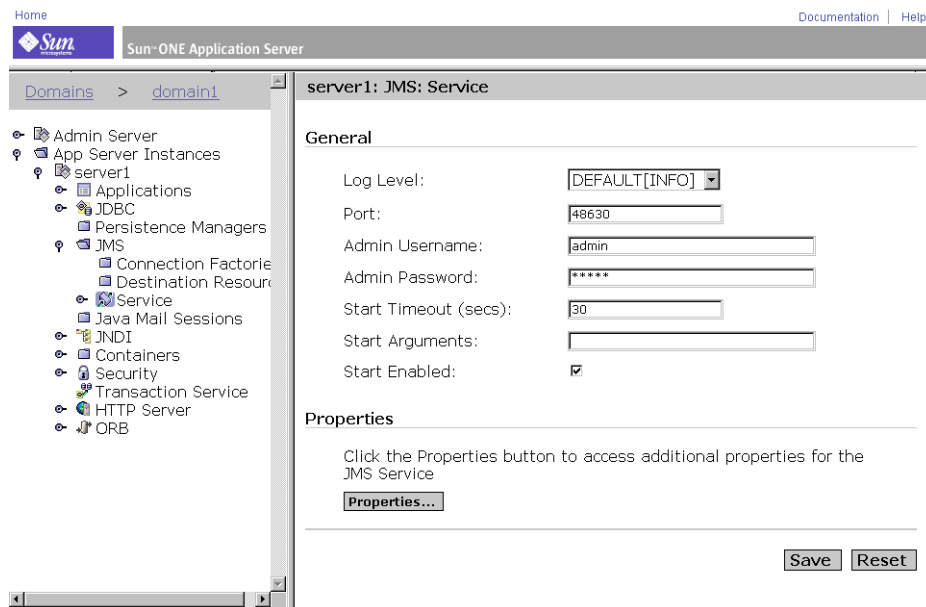
Logging can be enabled or disabled selectively at the subsystem level. Logging control for each subsystem is specified in the `server.xml` file, as described in the *Sun ONE Application Server Configuration File Reference*. Each subsystem has its own logger in accordance with the requirements of the JDK1.4 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 ONE Application Server

Subsystem	Element
Administration Server	<code><admin-service></code>
EJB Container	<code><ejb-container></code>
Web Container	<code><web-container></code>
MDB Container	<code><mdb-container></code>
Sun ONE Message Queue (JMS service)	<code><jms-service></code>
Security Service	<code><security-service></code>
Java Transaction Service (JTS)	<code><transaction-service></code>
Object Request Broker (ORB)	<code><iiop-service></code>
Default handlers ¹	<code><log-service></code>

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.

Figure 5-2 Log Level for JMS Service

The table “[Log Levels](#)” on [page 98](#) defines the log levels, in increasing order of severity, provided for messages in Sun ONE Application Server. These log levels match the requirements of the JDK1.4 logging API specification. In addition, the log levels ALERT and FATAL are specific to Sun ONE Application Server and are not supported in the JDK1.4 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
<code>log-service</code>	<code>file</code>	ACC log file; when empty or missing, log to <code>stdout</code> .
<code>log-service</code>	<code>level</code>	ACC log level.

An example of the `sun-acc.xml` file is provided in the *Sun ONE 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 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 ONE Application Server Configuration File Reference* for more information on the use of `stdout` and `stderr`.

See [“Configuring the Log Service” on page 112](#) 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 ONE 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 ONE 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`”.

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:

- [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 106.

For Solaris (any version)

- Using Solaris `cron` utility. For more information, see “[Rotation Using Solaris “cron” Utility](#)” on page 108.

Internal-daemon Log Rotation

Internal-daemon log rotation is available for UNIX operating system. Internal-daemon log rotation occurs within the HTTP daemon and can only be configured at the server instance startup time. 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 ONE Application Server creates a time-stamped access log file and rotation starts at server startup.

Once the rotation starts, Sun ONE 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 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 ONE 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 110](#).

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 One 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 ld -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`.

NOTE When multiple options are specified at a time, there is an implied **AND** 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. This can be done by using 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  
s
```

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

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

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

```
/opt/SUNWappserver7/appserver/domains/domain1/mainserver/bin/rotate  
logs
```

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 114](#).

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/apps
erv/domains/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 ONE 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](#)

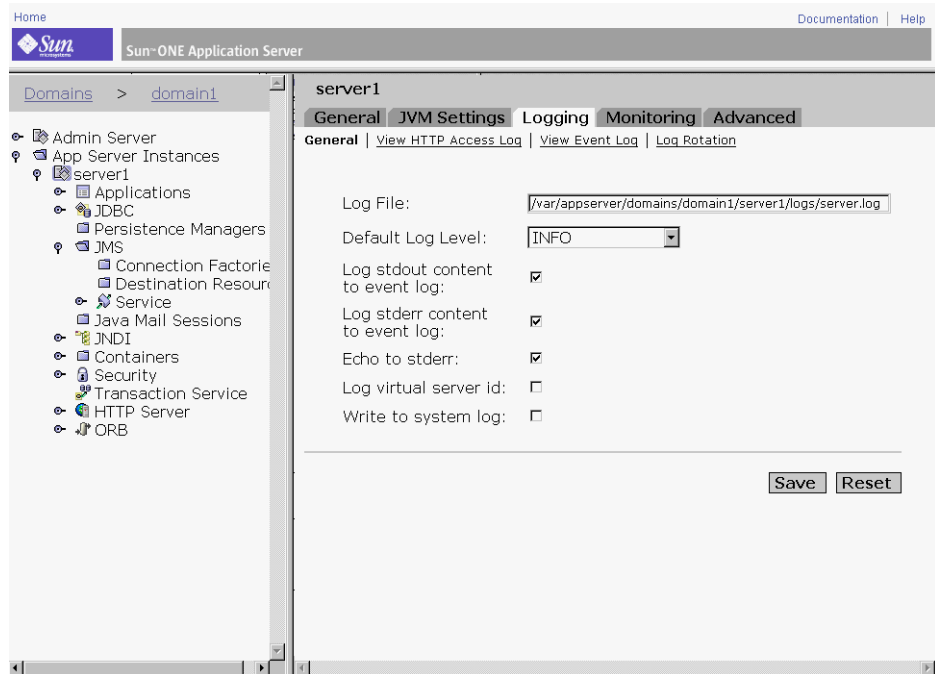
- [Configuring Logging for Application Server Components and Subsystems](#)
- [Configuring the Directives for Error Logging](#)

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 ONE 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.

Figure 5-3 Log Service Administration for the Service Instance

You can configure the following attributes for the log service element through the Administration interface, as shown in the figure “[Log Service Administration for the Service Instance](#)” on page 113.

- 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 ¹	(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	<p>(optional) Controls the default type of messages logged by other elements to the server log. Allowed values are as follows, from highest to lowest: <code>FINEST</code>, <code>FINER</code>, <code>FINE</code>, <code>CONFIG</code>, <code>INFO</code>, <code>WARNING</code>, <code>SEVERE</code>, <code>ALERT</code>, <code>FATAL</code>.</p> <p>Each value logs all messages for all lower values; for example, <code>FINEST</code> logs all messages, and <code>FATAL</code> logs only <code>FATAL</code> messages. The default value is <code>INFO</code>, which logs all <code>INFO</code>, <code>WARNING</code>, <code>SEVERE</code>, <code>ALERT</code>, and <code>FATAL</code> messages.</p>
log-stdout	True	(optional) If <code>true</code> , redirects <code>stdout</code> output to the server log. Legal values are <code>on</code> , <code>off</code> , <code>yes</code> , <code>no</code> , <code>1</code> , <code>0</code> , <code>true</code> , <code>false</code> .
log-stderr	True	(optional) If <code>true</code> , redirects <code>stderr</code> output to the server log. Legal values are <code>on</code> , <code>off</code> , <code>yes</code> , <code>no</code> , <code>1</code> , <code>0</code> , <code>true</code> , <code>false</code> .
echo-log-messages-to-stderr	True	(optional) If <code>true</code> , sends log messages to <code>stderr</code> in addition to the server log. Legal values are <code>on</code> , <code>off</code> , <code>yes</code> , <code>no</code> , <code>1</code> , <code>0</code> , <code>true</code> , <code>false</code> .
create-console	False	(optional) If <code>true</code> , creates a console window on Windows operating system for <code>stderr</code> output. Legal values are <code>on</code> , <code>off</code> , <code>yes</code> , <code>no</code> , <code>1</code> , <code>0</code> , <code>true</code> , <code>false</code> .
log-virtual-server-id	False	(optional) If <code>true</code> , virtual server IDs are displayed in the virtual server logs. These are useful if multiple <code>virtual-server</code> elements share the same log file.
use-system-log	False	If <code>true</code> , uses the UNIX <code>syslog</code> service 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 ONE 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 ONE 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 96](#).

To Specify a Log File: (Virtual Server)

To specify a log file, perform the following procedure:

1. In the left pane of the Administration interface, expand the Sun ONE 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 page 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)

To specify a transaction log location, perform the following procedure:

1. In the left pane of the Administration interface, expand the Sun ONE Application Server instance to display the Transaction Service subsystem.
2. Click the Transaction Service link.
3. In the right page 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 ONE 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`.

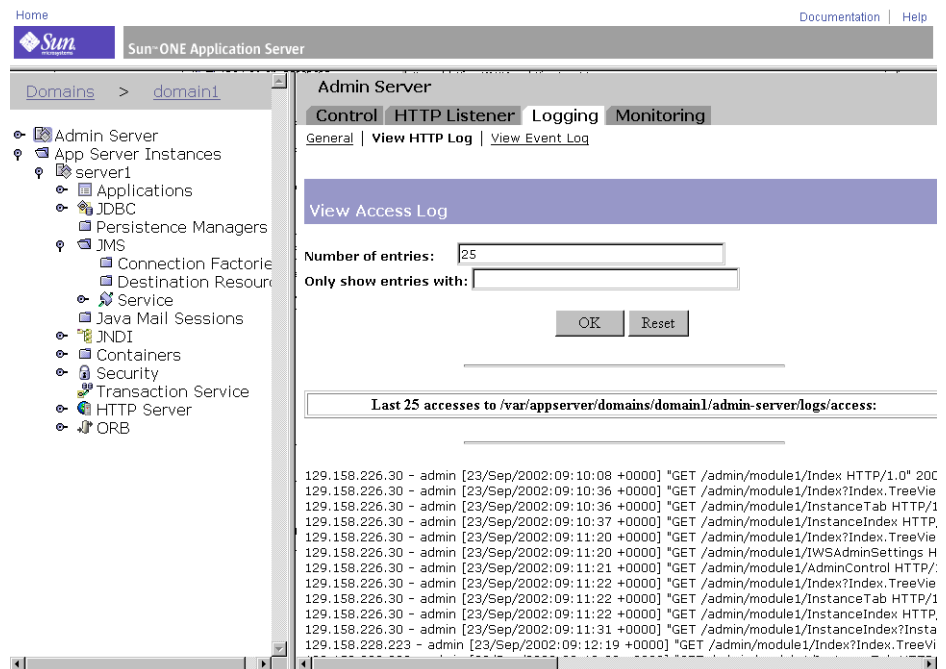
All the directives for `init.conf` are fully described in the *Sun ONE Application Server Configuration File Reference*.

Viewing the Access Log File

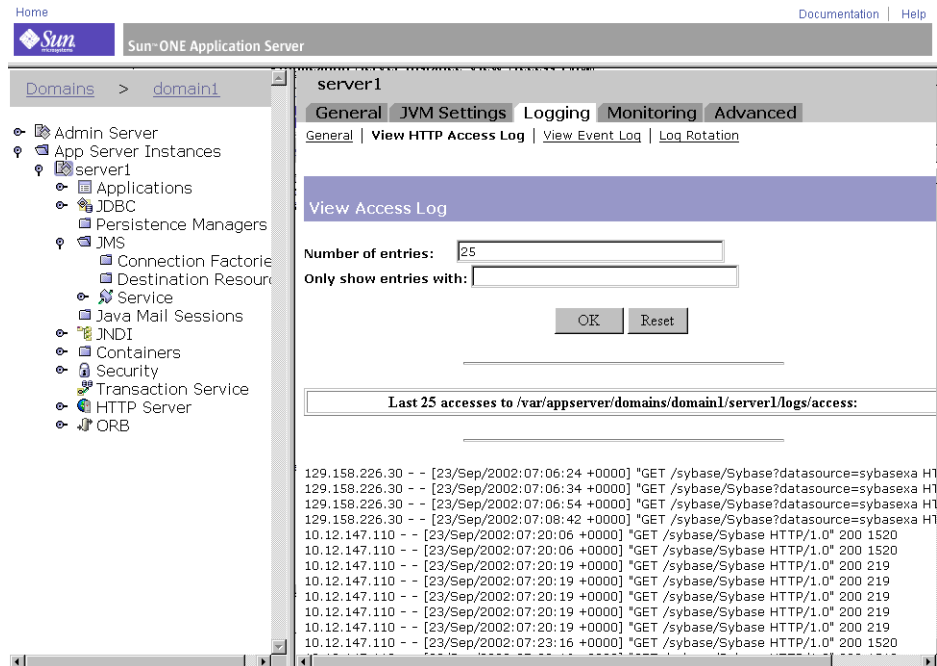
You can view both the Administrator server's and the Sun ONE Application Server instance's http log files.

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. An example of the displayed log is shown in the figure “Admin Server View HTTP Access Log” on page 117.

Figure 5-4 Admin Server View HTTP Access Log



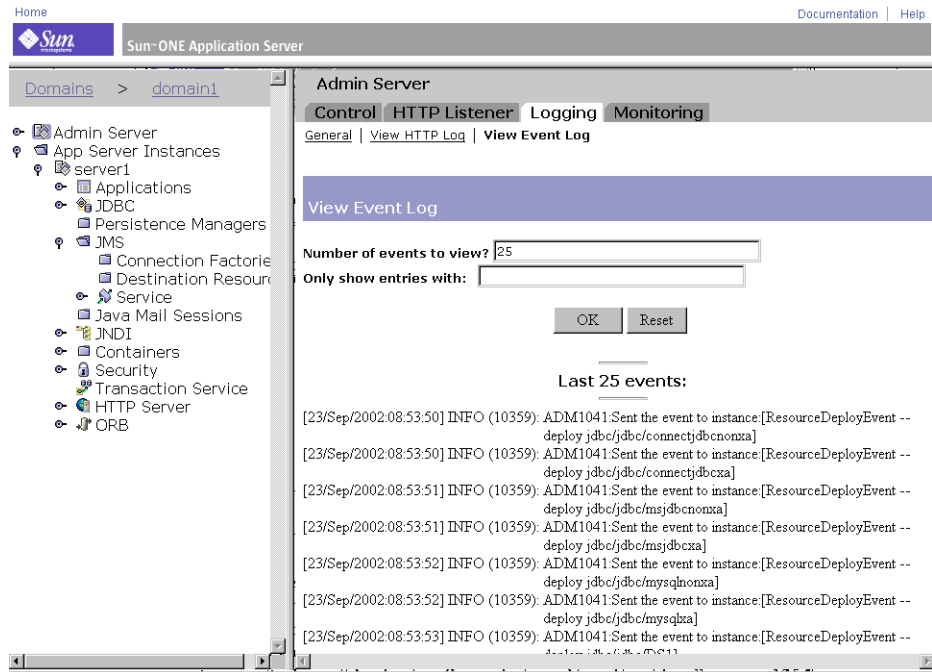
To view an access log for the application server instance, click the server instance desired from the left pane of the Administration interface. Click on the Logging tab from the right pane. Click the View Access Log link to display the configured active access log for that server instance. An example is shown in the figure “Application Server Instance View Access Log” on page 118.

Figure 5-5 Application Server Instance View Access Log

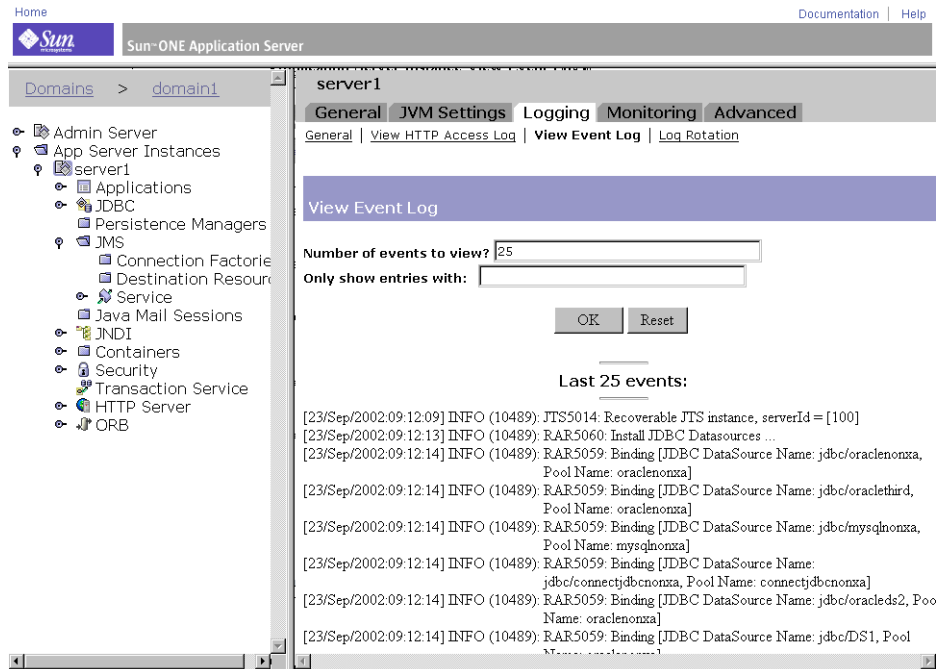
Viewing the Event Log File

You can view both the Administrator server's and the Sun ONE 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. An example of the displayed log is shown in the figure "[Admin Server View Event Log](#)" on page 119.

Figure 5-6 Admin Server View 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. An example of the displayed log is shown in the figure “[Application Server Instance View Event Log](#)” on page 120.

Figure 5-7 Application Server Instance View 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 ONE 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 ONE Application Server.

This setting requires a restart of the Sun ONE 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 117](#).

NOTE Before running the log analyzer, you must rotate the server logs. For more information, see [“Log File Management” on page 103](#).

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/SUNQappserver7/domains/domain1/server1/logs/access

```

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

Monitoring Sun ONE Application Server

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

This chapter includes the following sections:

- [About Monitoring Sun ONE 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 ONE Application Server

You can monitor Sun ONE 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 application server instance. Either the `asadmin` utility or SNMP can be used to monitor the Sun ONE Application Server 7, Enterprise Edition.

The following topics are addressed in this section:

- [Statistics](#)

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

Statistics

Statistics collection is always enabled for most of the Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition Performance Tuning and Sizing Guide*.

SNMP

The Sun ONE 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 158](#) and [“Setting Up SNMP” on page 166](#).

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 142](#) and [“Monitorable HTTP Server Attributes” on page 143](#).

NOTE	Only HTTP server statistics are available using SNMP. Statistics for all the subsystems of the Sun ONE Application Server 7, Enterprise Edition, 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 547](#).

Application Components and Subsystems Monitoring

Some of the subsystems or components within the Sun ONE Application Server 7, Enterprise Edition 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 137](#).

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 151](#) for further information.

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 ONE 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 151](#).

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 151](#).

This section addresses the following topics:

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

The `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 133](#).

The 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 132](#) 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.iio-service.orb.system.orb-connection.bad-name

Could not get the attribute
```

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

CLI Name Mapping

The Sun ONE 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 135](#).

The root object in the tree is represented by the Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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-m
ethod.getUserName3
```

Output is:

```
No monitorable entities for element
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB.bean-m
ethod.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.execution-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 Instance Name column describes the possible namespace.

The following table describes the tree structure in terms of possible parent-child relationships among various node types and the namespaces for some of the node types.

Table 6-1 Monitoring Object Types

Node Type	Singleton?	Leaf?	Child Node Types	Instance Names
root	Yes	No	http-server iiop-service resources transaction-service application standalone-ejb-module	
http-server	Yes	No	virtual-server process	
virtual-server	Yes	Yes		
process	Yes	Yes		
iiop-service	Yes	Yes	orb	
orb	No	No	orb-connection orb-thread-pool	system is reserved for system ORB. All user ORBs get a name derived from TCP endpoint.
orb-connection	Yes	Yes		
orb-thread-pool	Yes	Yes		
resources	Yes	No	jdbc-connection-pool	
jdbc-connection-pool	No	Yes		The names are the same as those specified by the user while creating a connection pool.
transaction-service	Yes	Yes		
application	No	No	ejb-module	Name of the application as registered in server.xml.
ejb-module	No	No	stateless-session-bean stateful-session-bean entity-bean message-driven-bean	Name of the EJB module. It is derived from the EJB JAR name.

Table 6-1 Monitoring Object Types (*Continued*)

Node Type	Single-ton?	Leaf?	Child Node Types	Instance Names
standalone-ejb-module	No	No	stateless-session-bean stateful-session-bean entity-bean message-driven-bean	Name of the standalone EJB module as registered in <code>server.xml</code> .
stateless-session-bean	No	No	bean-pool bean-method	Name of the bean from the deployment descriptor.
stateful-session-bean	No	No	bean-cache bean-method	Name of the bean from the deployment descriptor.
entity-bean	No	No	bean-cache bean-pool bean-method	Name of the bean from the deployment descriptor.
message-driven-bean	No	No	bean-pool bean-method	Name of the bean from the deployment descriptor.
bean-pool	Yes	Yes		
bean-cache	Yes	Yes		
bean-method	No	Yes		<code>onMessage</code> for message-driven beans, method name followed by a numeric suffix for methods in other enterprise beans. (The suffix is needed to disambiguate overloaded methods.)

Monitable Attribute Names

It is not necessary for every monitorable object to expose monitorable attributes. Some of the objects are used just for grouping other objects. For the Sun ONE Application Server 7, Enterprise Edition, 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 monitorable attribute names for various nodes.

Table 6-2 http-server

Attribute Name	Datatype	Description
summary	String (Formatted)	HTTP server summary. Includes virtual servers and processes. NOTE: See section “HTTP Server Monitorable Objects” on page 142 for more information on what data is exposed in the formatted string.

Table 6-3 virtual-server

Attribute Name	Datatype	Description
<vs-id>	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 summary 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.<vs-id></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 “HTTP Server Monitorable Objects” on page 142 for more information on what data is exposed in the formatted string.

Table 6-4 process

Attribute Name	Datatype	Description
<pid>	String (Formatted)	Process information. For every application server instance there is one process. The process ID can be obtained from the summary attribute of <code>http-server</code> . Statistics for a process can be obtained using the parameter of the form <code>server1.http-server.process.<pid></code> to the <code>get</code> command. NOTE: See section “HTTP Server Monitorable Objects” on page 142 for more information on what data is exposed in the formatted string.

Table 6-5 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 6-6 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 6-7 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 6-8 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 6-9 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 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 6-10 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.

Table 6-10 bean-cache (Continued)

Attribute Name	Datatype	Description
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.
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 6-11 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.

Table 6-11 bean-method (Continued)

Attribute Name	Datatype	Description
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 ONE Application Server 7, Enterprise Edition Developer's Guide to NSAPI*.

For information on how to use the monitoring statistics for performance tuning, see the *Sun ONE Application Server 7, Enterprise Edition Performance and Tuning Guide*.

Monitorable HTTP Server Elements

The following table lists the HTTP server monitorable elements.

Table 6-12 Monitorable HTTP Server Elements

Element Name	Subelements	Description
Server	ConnectionQueue ThreadPool Profile Process VirtualServer	A server instance.

Table 6-12 Monitorable HTTP Server Elements (*Continued*)

Element Name	Subelements	Description
ConnectionQueue	None	The queue in which requests are held prior to being serviced. There is only one connection queue in Sun ONE Application Server 7.
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	ConnectionQueueBucket ThreadPoolBucket DnsBucket DnsBucket KeepaliveBucket CacheBucket Thread	A single server process within a server instance.
ConnectionQueueBucket	None	Tracks statistics pertaining to a specific ConnectionQueue.
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 6-13 Server

Attribute Name	Values	Description
Id		The server instance ID (for example, <code>server1</code>).
VersionServer		A string that contains the Sun ONE Application Server 7, Enterprise Edition 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).
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 6-14 ConnectionQueue

Attribute Name	Values	Description
Id		The connection queue ID.

Table 6-15 ThreadPool

Attribute Name	Values	Description
Id		The thread pool ID.
Name		The symbolic name of the thread pool.

Table 6-16 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 6-17 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 6-18 ConnectionQueueBucket

Attribute Name	Values	Description
ConnectionQueue		The ID of a ConnectionQueue 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 CountTotalQueued may be greater than or equal to CountTotalConnections.
TicksTotalQueued		The total number of ticks that connections have spent in the queue. A tick is a system-dependent unit of time; see TicksPerSecond.

Table 6-19 ThreadPoolBucket

Attribute Name	Values	Description
Thread-pool		The ID of a ThreadPool 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.

Table 6-19 ThreadPoolBucket (*Continued*)

Attribute Name	Values	Description
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 6-20 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 6-21 KeepaliveBucket

Attribute Name	Values	Description
CountConnections		The number of connections currently in keepalive mode.
MaxConnections		The maximum number of simultaneous keepalive connections.

Table 6-21 KeepaliveBucket *(Continued)*

Attribute Name	Values	Description
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 6-22 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.
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.

Table 6-22 CacheBucket (*Continued*)

Attribute Name	Values	Description
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 6-23 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 ConnectionQueue the thread is servicing.

Table 6-24 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 6-25 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 6-26 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 following table describes the attributes that can be monitored to gather statistics for the JTS. These attributes can be set from the command line according to the rules described in [“CLI Name Mapping” on page 132](#).

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)

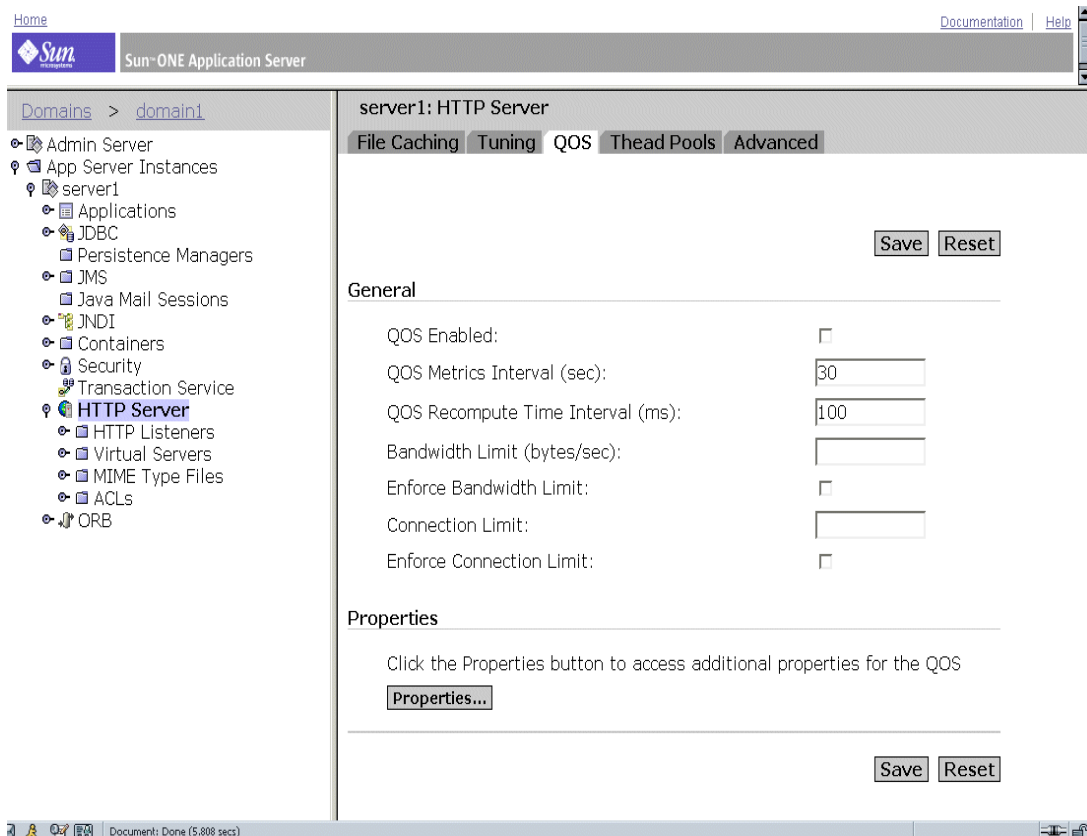
Quality of Service for a server instance or a class of virtual servers is configured through the Administration interface.

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 156](#).

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 following page displays the general settings for Quality of Service, followed by a properties button.

Figure 6-1 Virtual Server Instance QOS Tab

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 ONE 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 ONE 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 ONE 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 ONE 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 ONE Application Server.

NOTE	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 ONE Application Server 7, Enterprise Edition.

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 ONE 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 ONE 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 ONE 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 ONE Application Server has its own MIB located at *install_dir/lib*

The Sun ONE 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 ONE Application Server.

The Sun ONE 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 ONE Application Server 7, Enterprise Edition and monitor the server in real time using the Sun ONE Application Server MIB. The following table lists and describes the managed objects stored in the `appserv.mib` file.

`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>	Sun ONE Application Server 7, Enterprise Edition CPUs.
<code>iwsCpuUserTime</code>	CPU user time.
<code>iwsInstanceTable</code>	Sun ONE Application Server 7, Enterprise Edition instances.
<code>iwsInstanceId</code>	Server instance identifier
<code>iwsInstanceVersion</code>	String, such as SunONE-ApplicationServer-Enterprise/7 BB1-01/24/2001 17:15 (SunOS DOMESTIC)
<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.

appserv.mib Managed Objects and Descriptions (*Continued*)

Managed object	Description
iwsInstanceCount3xx	Number of 300-level (Redirection) responses issued by the server instance.
iwsInstanceCount4xx	Number of 400-level (Client Error) responses issued by the server instance.
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.

appserv.mib Managed Objects and Descriptions (*Continued*)

Managed object	Description
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.
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	Sun ONE 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.

appserv.mib Managed Objects and Descriptions (*Continued*)

Managed object	Description
<code>iwsProcessConnectionQueueMax</code>	Maximum number of connections allowed in connection queue.
<code>iwsProcessConnectionQueueTotal</code>	Number of connections that have been accepted.
<code>iwsProcessConnectionQueueOverflows</code>	Number of connections rejected due to connection queue overflow.
<code>iwsProcessKeepaliveCount</code>	Number of connections currently in keepalive queue.
<code>iwsProcessKeepaliveMax</code>	Maximum number of connections allowed in keepalive queue.
<code>iwsProcessSizeVirtual</code>	Process size in kilobytes.
<code>iwsProcessSizeResident</code>	Process resident size in kilobytes.
<code>iwsProcessFractionSystemMemoryUsage</code>	Fraction of process memory in system memory.
<code>iwsListenTable</code>	Sun ONE Application Server listen sockets.
<code>iwsListenId</code>	Listen socket identifier.
<code>iwsListenAddress</code>	Address where socket listens.
<code>iwsListenPort</code>	Port where socket listens.
<code>iwsListenSecurity</code>	Encryption support.
<code>iwsThreadPoolCount</code>	Number of requests denied.
<code>iwsThreadPoolMax</code>	Maximum number of requests allowed in queue.
<code>iwsThreadPoolPeak</code>	Largest number of requests that have been queued simultaneously.
<code>iwsThreadPoolTable</code>	Sun ONE Application Server thread pools.
<code>iwsVsCount503</code>	Number of 503 (Unavailable) responses issued.
<code>iwsInstanceCount503</code>	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 ONE 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 169](#) 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 169](#) 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 169](#).

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.

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">1. Start the master agent.2. Enable the subagent for each server installed on the system.
<ul style="list-style-type: none">• Native agent is currently running• No SMUX• No need to continue using native agent	<ol style="list-style-type: none">1. Stop the native agent when you install the master agent for your Administration Server.2. Start the master agent.3. Configure the SNMP subagent for each server instance.

If your server meets these conditions...	...follow these procedures. (discussed in detail in the following sections).
<ul style="list-style-type: none"> Native agent is currently running No SMUX Needs to continue using native agent 	<ol style="list-style-type: none"> 1. Install a proxy SNMP agent. 2. Start the proxy SNMP agent. 3. Restart the native agent using a port number other than the master agent port number. 4. Start the master agent. 5. Enable the subagent for each server installed on the system.

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.

NOTE

After changing SNMP settings in the Admin Server, installing a new server, or deleting an existing server, you must perform the following steps:

- (UNIX) Restart the SNMP master agent and the SNMP subagent using the Admin Server.

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 ONE Application Server master agent. Before you start, be sure to stop the native master agent. (See your system documentation for detailed information.)

NOTE	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 ONE 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 169](#).
2. Install and start the proxy SNMP agent and restart the native SNMP daemon. See [“Using a Proxy SNMP Agent” on page 167](#).
3. Start the SNMP master agent. See [“Enabling and Starting the SNMP Master Agent” on page 172](#).
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

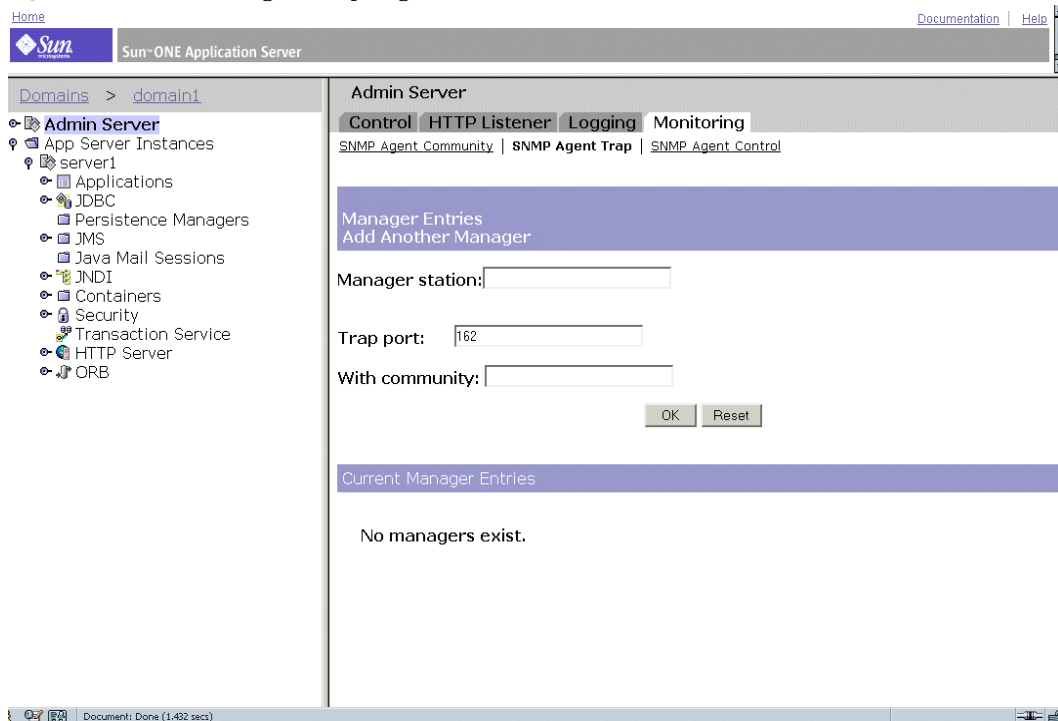
NOTE You cannot use the Administration interface to install and start the master SNMP agent unless the server is running as `root`.

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, as shown in the following figure:

Figure 6-2 SNMP Agent Trap Page



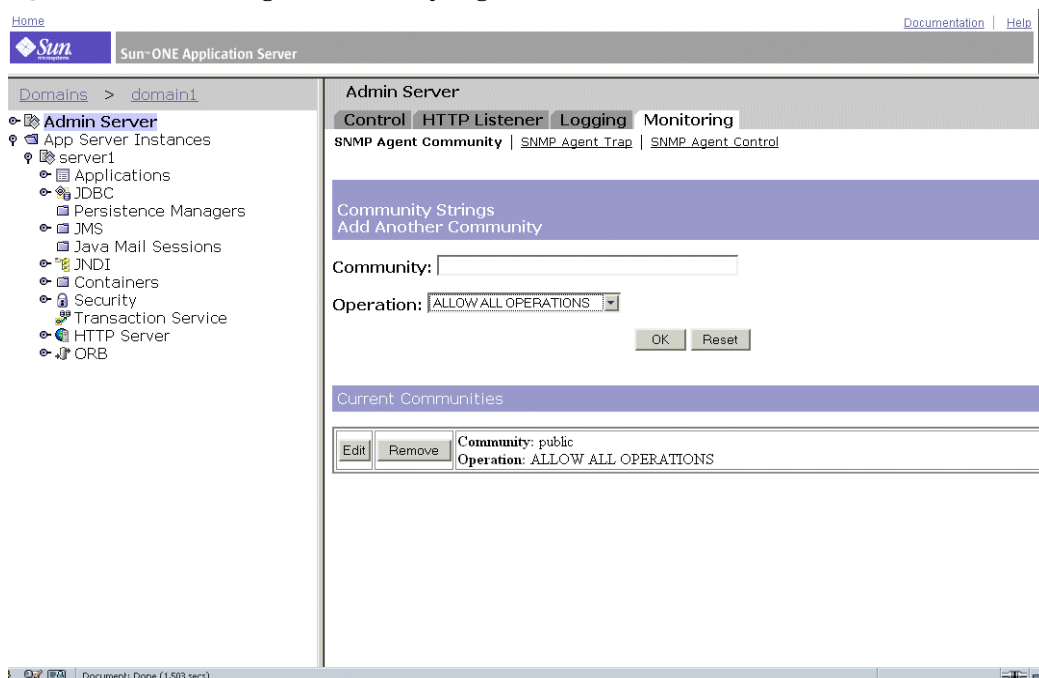
The Manager Entries information is displayed on this page.

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 165](#).
8. Type the community string you want to use in the trap. For more information on community strings, see [“SNMP Agent Community” on page 166](#).
9. Click OK.
10. Click the SNMP Agent Community link within the Monitoring tab.

The Community Strings information is displayed, as shown in the following figure.

Figure 6-3 SNMP Agent Community Page



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.

NOTE	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 <i>pid</i></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 Master Agent
- 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 174](#).

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.

The following page is displayed:

Figure 6-4 SNMP Subagent Configuration Page

Home [Documentation](#) [Help](#)

Sun ONE Application Server

Domains > domain1

- Admin Server
- App Server Instances
 - server1**
 - Applications
 - JDBC
 - Persistence Managers
 - JMS
 - Java Mail Sessions
 - JNDI
 - Containers
 - Security
 - Transaction Service
 - HTTP Server
 - ORB

server1

General **JVM Settings** **Logging** **Monitoring** **Advanced**

SNMP Subagent Configuration | [SNMP Subagent Control](#)

SNMP Configuration

Master Host:

Description:

Organization:

Location:

Contact:

Enable SNMP Statistics Collection:

☒ Off

☐ On

Document: Done (1.473 sec)

- (UNIX only) Enter the name and domain of the server in the Master Host field.
- Enter the Description of the server, including operating system information.
- Enter the Organization responsible for the server.
- Enter the location of the server instance.
- Enter the name of the person responsible for the server and the person's contact information in the Contact field.
- Select On to Enable the SNMP Statistics Collection.
- Click OK.
- 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.

NOTE An invalid manager name in the `CONFIG` 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

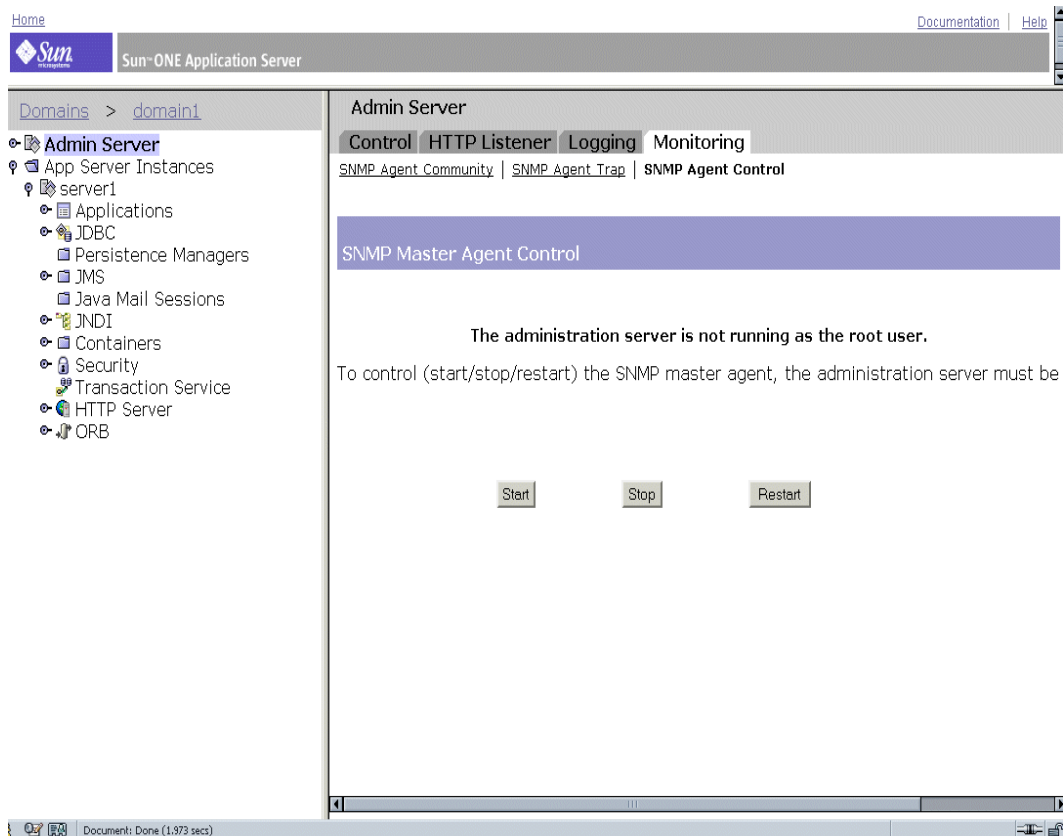
To start the SNMP master agent using the Admin Server, perform the following steps:

NOTE You must be logged in the Sun ONE 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 following page is displayed.

Figure 6-5 SNMP Agent Control Page



4. Click Start.

You can also stop and restart the SNMP master agent from the SNMP Agent Control page.

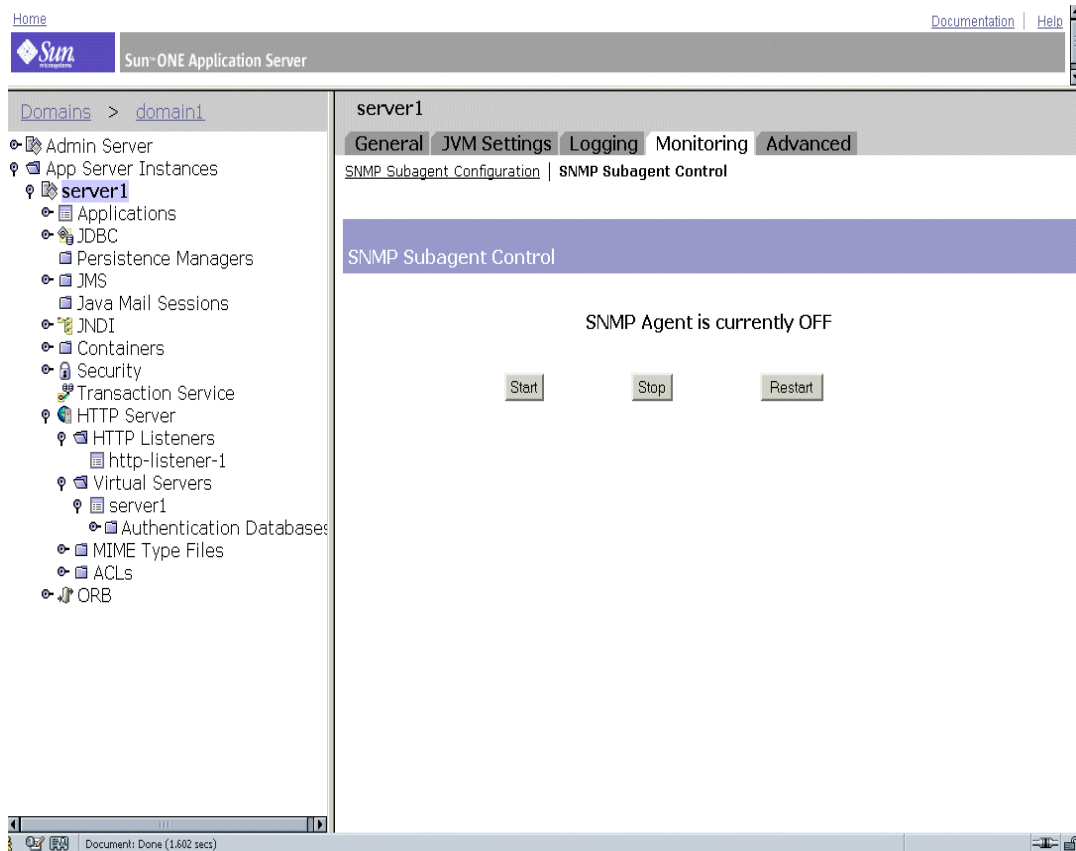
Enabling the Subagent

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 169](#).

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 page as shown in the following figure.

Figure 6-6 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.

NOTE

After making any SNMP configuration changes, you must click OK, then restart SNMP subagent from the SNMP Subagent Control page.

Configuring J2EE Containers

Sun ONE Application Server 7, Enterprise Edition 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 One 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

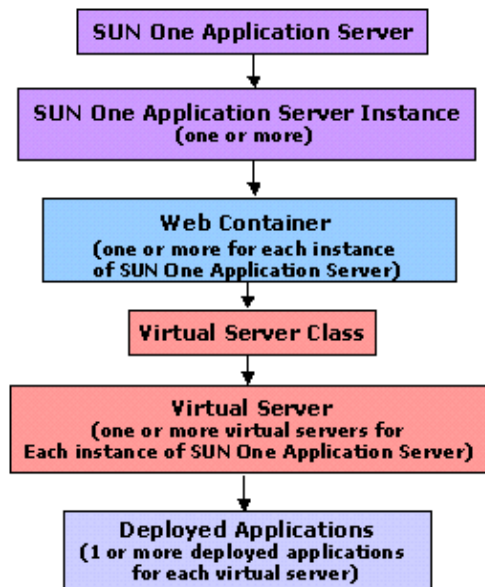
The Web Container is a J2EE container that hosts web applications. The web container extends the web server functionality by providing developers the environment to run servlets and Java Server Pages (JSPs). Servlets provide a component-based, platform-independent method for building web-based applications, 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).

The figure “[Web Container within the Sun ONE Application Server Architecture](#)” explains the role and the location of the web container, in the Sun ONE Application Server 7, Enterprise Edition architecture:

Figure 7-1 Web Container within the Sun ONE 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](#)

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 ONE Application Server 7, Enterprise Edition.

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 ONE Application Server 7, Enterprise Edition.

For more information on how to configure and use the web server plugin with Sun ONE Application Server 7, Enterprise Edition, see [“Configuring the Web Server Plugin” on page 187](#).

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 14, “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 186](#).

Web-module Attributes

The Sun ONE Application Server 7, Enterprise Edition-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 an `sun-web.xml` file. In the absence of the `sun-web.xml` file, the web container assumes default values for all Sun ONE Application Server 7, Enterprise Edition-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 re-deployment 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 Sun ONE Application Server 7, Enterprise Edition 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. You will see the page displayed in “[Logging the Web Container](#),” in the right pane of the Administration interface.

Figure 7-2 Logging the Web Container

server1: Containers: Web

General

Log Level:

Properties

Click the Properties button to access additional properties for the Web Container

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 the Properties button.

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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition.

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 ONE 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 ONE Application Server 7, Enterprise Edition.

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 ONE 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 Configuration](#)
- [Configuring EJB Settings](#)
- [Configuring MDB Pool Settings](#)

Performing General 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 Sun ONE Application Server 7, Enterprise Edition 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. You will see the window as shown in “[EJB Container - General Configuration](#)” in the right pane of the Administration interface.

Figure 7-3 EJB Container - General Configuration

server1: Containers: EJB Container

EJB Settings **MDB Settings**

General | [Default Pool Settings](#) | [Default Cache Settings](#)

Attributes

Monitoring Enabled: ☒

Log Level: DEFAULT[INFO] ▼

Commit Option: B ▼

Properties

Click the Properties button to access additional properties for EJBs

Properties...

Save **Reset**

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 ONE Application Server 7, Enterprise Edition. 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).
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 Sun ONE Application Server 7, Enterprise Edition 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. You will see the window as shown in “[Configuring EJB Pool Settings](#)” in the right pane of the Administration interface.

Figure 7-4 Configuring EJB Pool Settings

The screenshot shows the 'server1: Containers: EJB Container' configuration window. It has two tabs: 'EJB Settings' (selected) and 'MDB Settings'. Under 'EJB Settings', there are three sub-tabs: 'General', 'Default Pool Settings' (selected), and 'Default Cache Settings'. The 'Default Pool Settings' tab contains four input fields: 'Steady Pool Size' with value 32, 'Max Pool Size' with value 64, 'Pool Resize Quantity' with value 16, and 'Idle Timeout (secs)' with value 600. At the bottom right are 'Save' and 'Reset' buttons.

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 Sun ONE Application Server 7, Enterprise Edition 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. You will see the window as shown in “[Configuring EJB Pool Settings](#)” in the right pane of the Administration interface.

Figure 7-5 Configuring EJB Cache Settings

server1: Containers: EJB Container

EJB Settings | MDB Settings

General | Default Pool Settings | **Default Cache Settings**

Max Cache Size:

Cache Resize Quantity:

Removal Timeout (secs):

Victim Selection Policy:

Idle Timeout (secs):

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 Sun ONE Application Server 7, Enterprise Edition 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. You will see the window shown in “[Configuring MDB Pool Settings](#)” in the right pane of the Administration interface.

Figure 7-6 Configuring MDB Pool Settings

server1: Containers: EJB Container

EJB Settings MDB Settings

General | Default Pool Settings

Steady Pool Size:

Max Pool Size:

Pool Resize Quantity:

Idle Timeout (secs):

Save Reset

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.

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 ONE Application Server.

This chapter discusses Java Transactions in general, and the transaction support incorporated into Sun ONE Application Server 7, Enterprise Edition 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.

Sun ONE Application Server 7, Enterprise Edition 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 245](#). 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.

Sun ONE Application Server 7, Enterprise Edition comes integrated with Sun ONE Message Queue, a fully capable JMS provider and the corresponding transactional resource manager. In this manner, Sun ONE Application Server 7, Enterprise Edition enables transactional JMS access from servlets, JSP pages and enterprise beans. It is also possible to use third party JMS Providers with Sun ONE Application Server 7, Enterprise Edition. For more details, see [Chapter 10, “Using the JMS Service.”](#)

J2EE Connectors

Sun ONE Application Server 7, Enterprise Edition 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

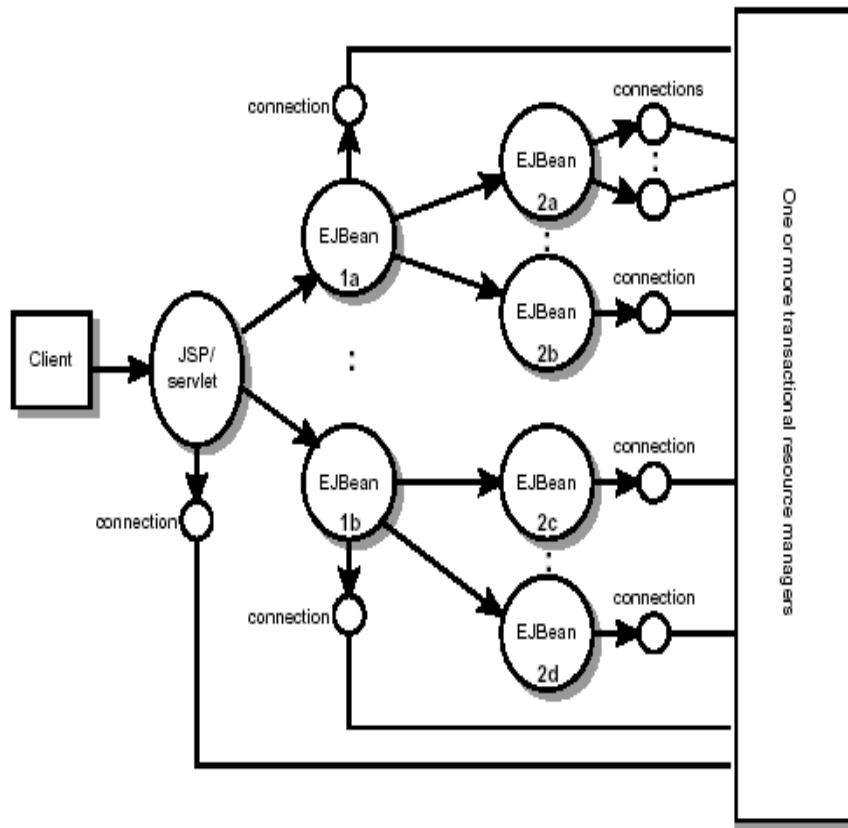
A transaction that involves only one resource can be completed using local transactions. A local transaction also requires that all participating application components execute within one process. Transactions that involve more than one resource, or multiple participant processes become distributed or 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 Sun ONE Application Server 7, Enterprise Edition, it may be necessary to provide additional configuration information in the Sun ONE Application Server 7, Enterprise Edition 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.

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 ONE 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 215](#).

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:

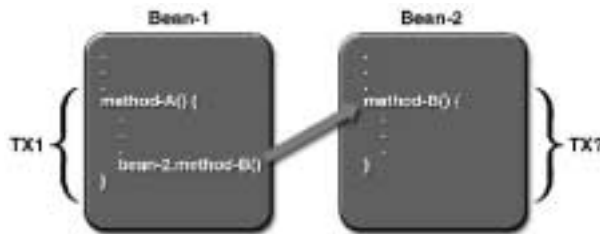
- [Transaction Attributes](#)

- [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 8-1 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

Table 8-1 Transaction Attributes

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 218](#).

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 Sun ONE Application Server 7, Enterprise Edition instance tree, whose transaction configuration you want to modify.
2. Select Transaction Service from the list of J2EE services that are displayed. You will see the following window, shown in the figure “Configuring Transaction Service Options” in the right pane of the Administration interface:

Figure 8-2 Configuring Transaction Service Options

General

Monitoring Enabled:

☒

Log Level:

INFO

Advanced

Recover on Restart:

☐

Response Timeout (secs):

0

Transaction Log Location:

/export/anh/as7/appserv/instances/austen3/logs

Heuristic Decision:

rollback

Keypoint Interval (txns):

2048

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

Table 8-2 Monitorable Attributes of Java Transaction Services

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)
txnidl1 Prepared 20
txnidl2 Active 100
txnidl3 Active 120
... ..
```

Managing Transactions

In the example given in [Listing In-Flight Transactions](#), 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)
txnid1 Prepared 500
txnid2 Active 360
```

Configuring Naming and Resources

This chapter describes the J2EE Resources used by Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition.

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 ONE Application Server 7, Enterprise Edition.

For more information on JDBC Datasources, see [“About JDBC Resources” on page 245](#).

Java Mail Sessions

JMS destinations are J2EE resources that can be created and managed via Sun ONE Application Server 7, Enterprise Edition.

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 ONE Application Server 7, Enterprise Edition. For more information on Java Mail Sessions, see [“About Java Mail Resources” on page 267](#).

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)

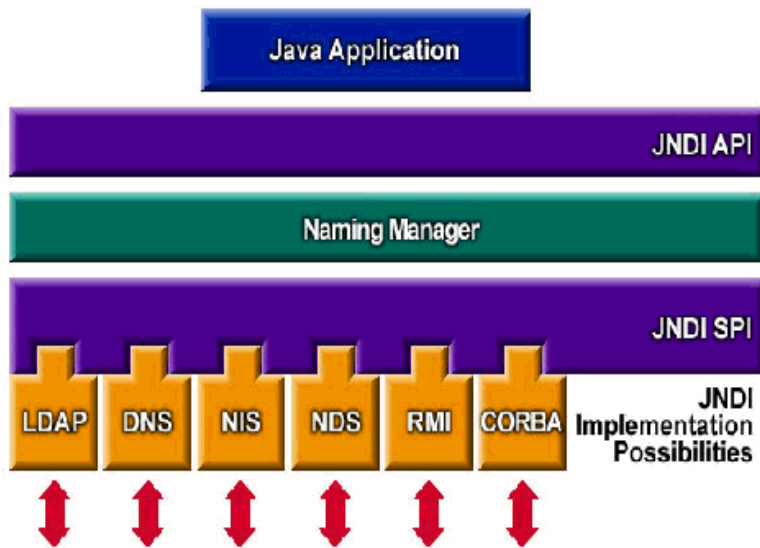
This section discusses the Java Naming and Directory Interface (JNDI). Java Naming and Directory Interface (JNDI) is an application programming interface (API) for accessing different kinds of naming and directory services. J2EE components locate objects by invoking the JNDI lookup method.

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. The following figure, "[Overview of the JNDI Architecture](#)," 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 ONE Application Server 7, Enterprise Edition 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 181](#).

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.

The following table, [“JNDI Lookups and Their Associated References”](#), lists JNDI lookups and their associated references for the J2EE resources used by Sun ONE Application Server 7, Enterprise Edition.

Table 9-1 JNDI Lookups and Their Associated References

JNDI Lookup Name	Associated Reference
<code>java:comp/env</code>	Application environment entries
<code>java:comp/env/jdbc</code>	JDBC DataSource resource manager connection factories
<code>java:comp/env/ejb</code>	EJB References
<code>java:comp/UserTransaction</code>	UserTransaction references
<code>java:comp/env/mail</code>	JavaMail Session Connection Factories
<code>java:comp/env/url</code>	URL Connection Factories
<code>java:comp/env/jms</code>	JMS Connection Factories and Destinations
<code>java:comp/ORB</code>	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 ONE Application Server 7, Enterprise Edition. 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. To know more about configuring Java Mail session resource factories, please see [“About Java Mail Resources” on page 267](#).

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 ONE Application Server 7, Enterprise Edition supports such lookup of the transaction co-ordinator. For more information about the `javax.transaction.UserTransaction` interface, see [“Using Transaction Services” on page 201](#).

Initial Naming Context

The naming support in Sun ONE Application Server 7, Enterprise Edition is based primarily on J2EE 1.3, with a few added enhancements. When an application component creates the initial context, via `InitialContext()`, Sun ONE Application Server 7, Enterprise Edition 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` namespace 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition server registers the entity beans for the IIOP clients to lookup and bind to.

Note that Sun ONE Application Server 7, Enterprise Edition 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 311.](#)

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 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 ONE Application Server 7, Enterprise Edition, 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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. You will see the [“JNDI Custom Resources Page”](#) in the right pane of the Administration interface:

Figure 9-2 JNDI Custom Resources Page

server1: JNDI: Custom Resources: New

JNDI Name: *

Resource Type: *

Factory Class: *

Description:

Custom Resource Enabled: ☒

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 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 ONE Application Server 7, Enterprise Edition 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.

You will see the following window, shown in the [“JNDI External Resources Page”](#) in the right pane of the Administration interface:

Figure 9-3 JNDI External Resources Page

server1: JNDI: External Resources: New

JNDI Name: *

Resource Type: *

JNDI Lookup: *

Factoryclass: *

Description:

External Resource Enabled: ☒

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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 </res-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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition-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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition.

This module covers the following topics:

- What is Persistence?
- The Role of the Persistence Manager

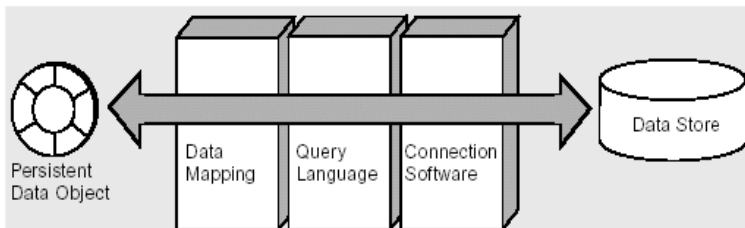
- Pre-Deployment Bean Configuration
- Creating a New Persistence Manager

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, “Basic Persistence Scheme”:

Figure 9-4 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

The 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.

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 ONE Application Server 7, Enterprise Edition-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 ONE Application Server 7, Enterprise Edition DTD:

PM descriptors contain one or more pm descriptors, but only of them must be in use at any given time

```
-->

<!ELEMENT pm-descriptors ( pm-descriptor+, pm-inuse)>
<!--

pm-descriptor describes the properties for the persistence manager
associated with entity bean

-->

<!ELEMENT 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 ONE Application Server 7, Enterprise Edition Transparent Persistence, TopLink, Versant or CocoBase:

```
-->

<!ELEMENT pm-identifier (#PCDATA)>
<!--

pm-version further specifies which version of PM vendor product to
be used

-->

<!ELEMENT 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition, you will see the list displayed in the right pane of the Administration interface.

2. To create a new Persistence Manager, click New. You will see the following window shown in the figure [“Creating a New Persistence Manager”](#):

Figure 9-5 Creating a New Persistence Manager

server1: Persistence Managers: New

General

JNDI Name:*

Description:

Factory Class:

Connection Pool:*

A JDBC resource will be automatically created to associate the Persistence Manager run-time with the specified Connection Pool.

Persistence Manager Enabled: ☒

* Indicates Required Field

3. 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 ONE Application Server 7, Enterprise Edition'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 ONE Application Server 7, Enterprise Edition, 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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*)>
<!-- ATTENTION: The following attributes are required -->
<!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 ONE Application Server 7, Enterprise Edition 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, shown in the figure “[Creating a New JDBC Resource](#),” appears in the right pane.

Figure 9-6 Creating a New JDBC Resource

server1: JDBC: JDBC Resources: New

JNDI Name:*

Pool Name:*

Description:

Data Source Enabled: ☒

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 255](#).
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:

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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition instance for which you want to create a JDBC connection pool.
2. Select JDBC from the list of J2EE services listed under your Sun ONE Application Server 7, Enterprise Edition, and open the `ConnectionPools` tab under it. You will see the figure “Creating a New JDBC Connection Pool” in the right pane of the Administration interface.

Figure 9-7 Creating a New JDBC Connection Pool

server1: JDBC: Connection Pools: New

General

Enter the Connection Pool name, select a Database Vendor, and click Next. Properties for the selected Database Vendor will be displayed

Name:*

Global Transaction Support: ☒ Enabled

Database Vendor:*

◀ Back Next ▶ Reset Cancel

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](#) as given in “[Creating a New JDBC Connection Pool Using the Administration Interface](#)” on page 255. 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"> • <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. • <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 <table-name>". 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.
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 ONE Application Server 7, Enterprise Edition 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
Application server admin user name	<i>admin</i>
Application server admin password	<i>adminadmin</i>
Application server admin port	<i>8888</i>
Application server machine name	<i>sas.sun.com</i>
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 the “[Options required for creating JDBC Connection Pools using the Command Line Interface](#)” table.

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]
[--isisolationguaranteed] [--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\:@oracleser
ver.sun.com\:1521\:ORA" SampleJdbcConnectionPool
```

NOTE	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 ONE Application Server 7, Enterprise Edition.

The following is the syntax for applying the changes to your instance of Sun ONE Application Server 7, Enterprise Edition.

```
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 ONE Application Server 7, Enterprise Edition.

```
asadmin reconfig --user admin --password adminadmin --host sas.sun.com
--port 8888 server1
```

Once the changes are applied to your instance of Sun ONE Application Server 7, Enterprise Edition, 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition.

```
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 256](#).

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
<code>total-threads-waiting</code>	Integer	Total threads waiting for JDBC connection.
<code>total-outbound-connections</code>	Integer	Total JDBC connection validation failures
<code>total-connections-timed-out</code>	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 ONE Application Server 7, Enterprise Edition 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 Sun ONE Application Server 7, Enterprise Edition Deployment Descriptor](#)
- [Creating a New JavaMail Session](#)
- [Configuring Advanced Resource Properties](#)

About the JavaMail Message-handling Process

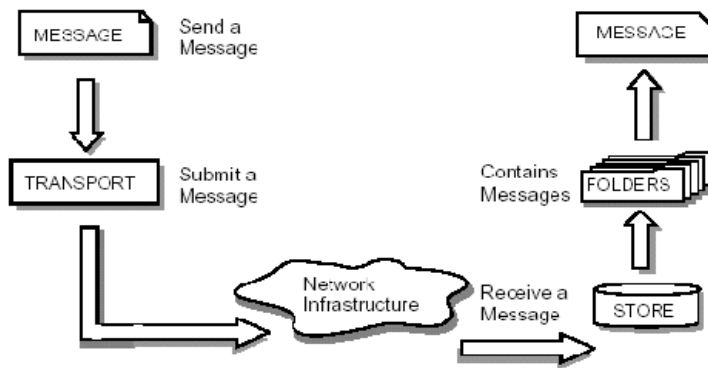
The JavaMail API performs the following functions, which comprise the standard mail handling process for a typical client application:

- 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.

NOTE 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 9-8 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 ONE Application Server 7, Enterprise Edition 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:

- [The Message Class](#)
- [Message Storage and Retrieval](#)
- [Message Composition and Transport](#)

The 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/ Message/rfc822	<code>javax.mail.internet.MIME.Multipart</code> <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 ONE Application Server 7, Enterprise Edition. 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 Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition specific xml. The fragment of the Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition instance, for which you want to create a new JavaMail session.
2. Click JavaMail Sessions. You will see the following window, shown in the figure “Configuring a JavaMail Session”, in the right pane of the Administration interface:

Figure 9-9 Configuring a JavaMail Session

server1: Java Mail Sessions: New

OK Cancel

General

JNDI Name:* NewJavaMailSession

Mail Host:* blr-root.india.sun.com

Default User:* ar126587

Default Return Address:* aruna.r@sun.com

Description: xyz

Java Mail Session Enabled: ☒

3. 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.
4. 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.
5. 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.
6. 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.
7. In the Description field, provide a description for this Java Mail session.
8. Mark the Java Mail Session Enabled checkbox to enable the Java Mail session that you created.
9. 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 ONE Application Server 7, Enterprise Edition instance, whose JavaMail session configuration you want to modify.
2. Click JavaMail Sessions. You will see the following window, shown in the figure “Configuring Additional Resources for JavaMail Session”, in the right pane of the Administration interface, below the main configuration section explained in “Creating a New JavaMail Session”:

Figure 9-10 Configuring Additional Resources for JavaMail Session

Advanced

Store Protocol:	<input type="text" value="imap"/>
Store Protocol Class:	<input type="text" value="com.sun.mail.imap.IMAPStore"/>
Transport Protocol:	<input type="text" value="smtp"/>
Transport Protocol Class:	<input type="text" value="com.sun.mail.smtp.SMTPTransport"/>
Debug Enabled:	<input type="checkbox"/>

3. 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.
4. 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.
5. In the Transport Protocol text field, specify the transport protocol that you want to use for this specific session of JavaMail, for example, SMTP.

6. 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.
7. Mark the Debug Enabled checkbox, to enable debugging of this specific session of JavaMail. Enabling this checkbox turns the debug mode on.
8. 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition support for JMS messaging, in general, and for MDBs, in particular, requires messaging middleware that implements the JMS specification—a JMS provider. Sun ONE Application Server 7, Enterprise Edition uses Sun ONE Message Queue (MQ), Version 3.01, as its native JMS provider.

MQ is tightly integrated into Sun ONE Application Server 7, Enterprise Edition, providing transparent JMS messaging support. This support (known within Sun ONE Application Server 7, Enterprise Edition 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 ONE Message Queue. It includes the following sections:

- [About JMS](#)
- [The 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 systems 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](#)
- [The 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.

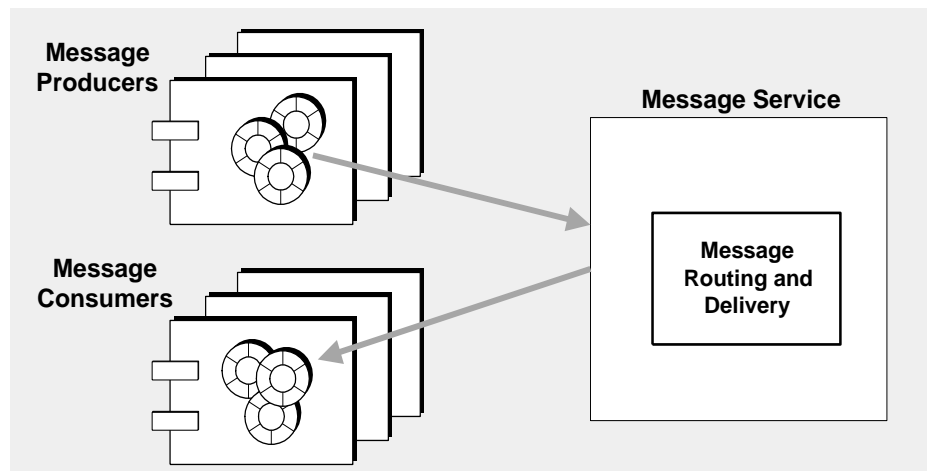
Message

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 the figure “[Message Service Architecture](#).” 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.

The 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. Properties are optional.

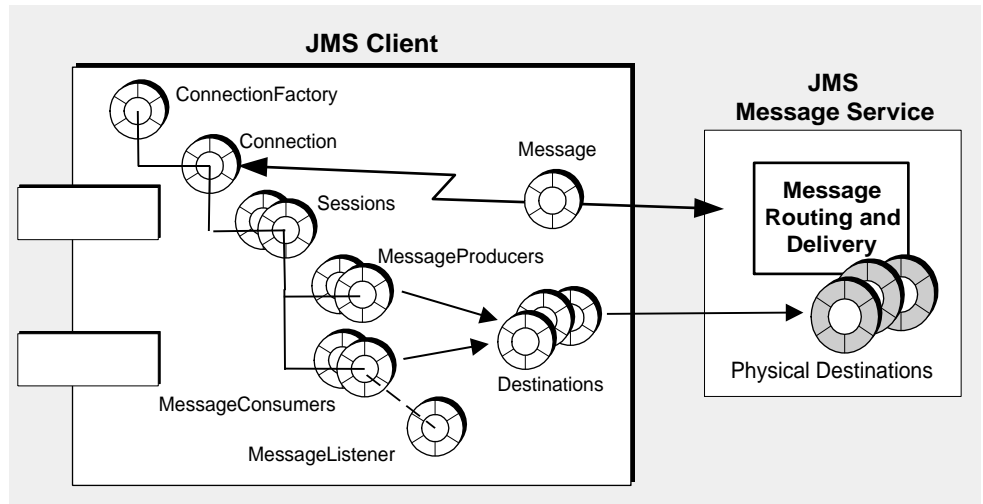
Message body. The message body contains the actual data to be exchanged. JMS supports six body types.

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. The figure “[JMS Programming Objects](#)” 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 the figure “[JMS Programming Model](#)” on page 282 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.

NOTE	In the context of the Sun ONE Application Server 7, Enterprise Edition, JMS administered objects are regarded as JMS resources, similar to other Application Server resources.
-------------	--

Message-driven Beans

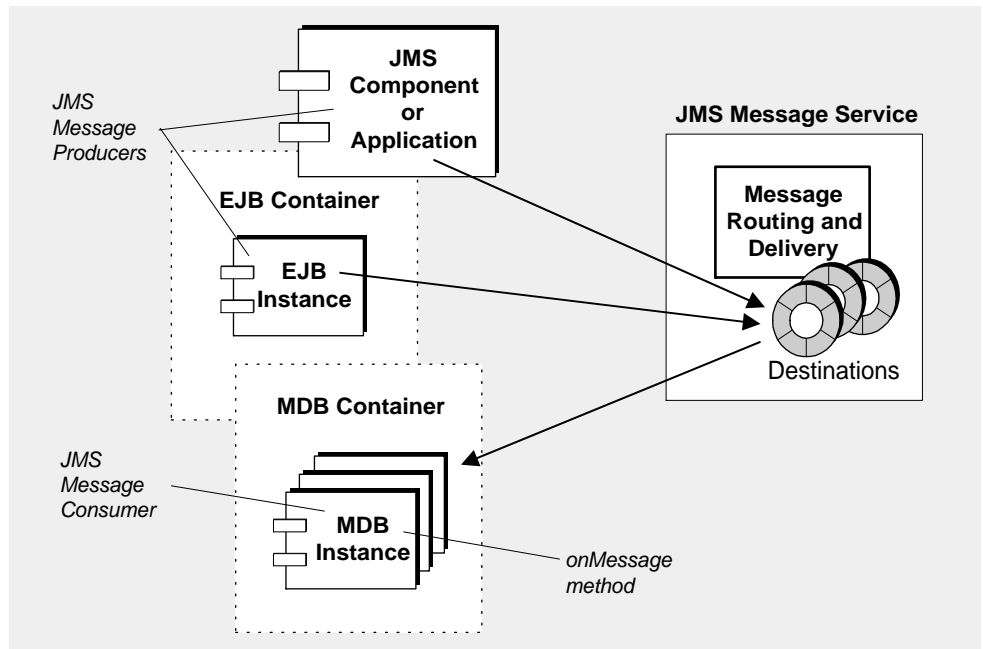
In addition to the general JMS client programming model introduced in the figure “[JMS Programming Model](#)” on page 282, there is a more specialized adaptation of the JMS API used in the context of J2EE applications. This specialized JMS client is called a *message-driven bean* and is one of a family 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 implements 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 the “[MDB Message Consumers](#)” figure.

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 ONE Application Server 7, Enterprise Edition, see “[About Message-driven Beans](#)” on page 193.

The Built-in JMS Service

Support for JMS messaging, in general, and for MDBs, in particular, is built into Sun ONE Application Server 7, Enterprise Edition. This support is achieved through the tight integration of Sun ONE Message Queue with the Sun ONE Application Server 7, Enterprise Edition, providing a native, built-in JMS Service.

This topic covers the following topics necessary to understand this built-in JMS Service:

- [About Sun ONE Message Queue \(MQ\)](#)
- [Integration of MQ with Sun ONE Application Server 7, Enterprise Edition](#)

For information on administering the built-in JMS Service, see “[Administration of the Built-in JMS Service](#)” on page 295.

About Sun ONE Message Queue (MQ)

Sun ONE 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 ONE Application Server 7, Enterprise Edition:

- 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 Sun ONE 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 the figure [“MQ System Architecture” on page 289](#), 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 MQ *Administrator's Guide*, which can be found at the following location:

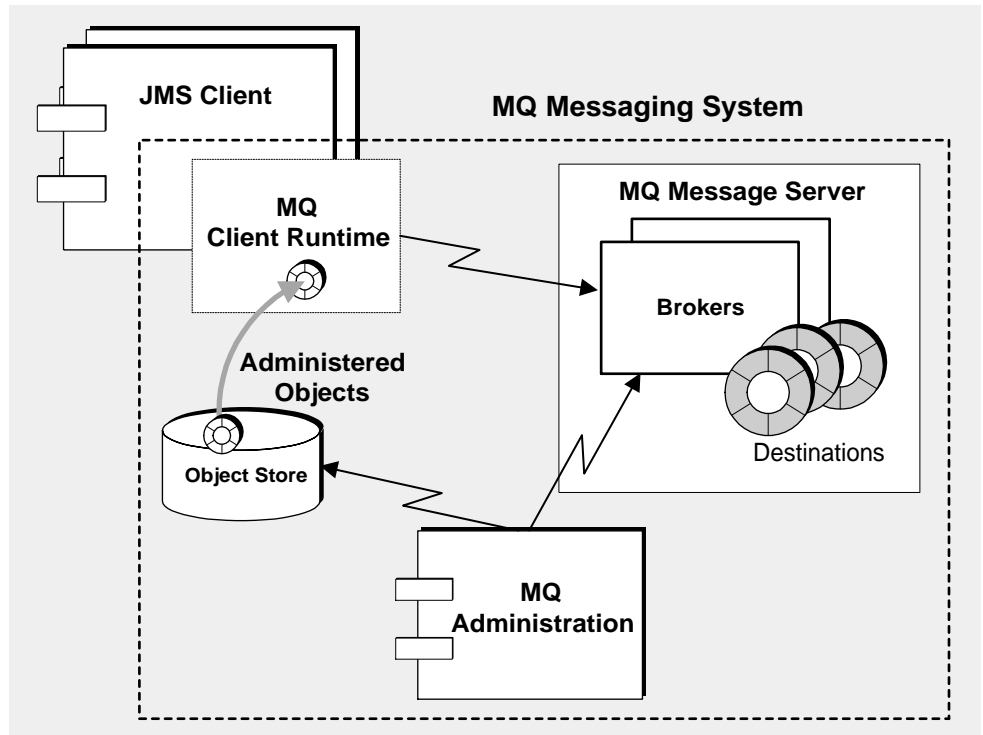
<http://docs.sun.com/>

MQ Message Server

The main parts of an MQ message server, as shown in the figure “[MQ System Architecture](#)” on page 289, 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 290 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 MQ 3.01 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 Administrator's 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.

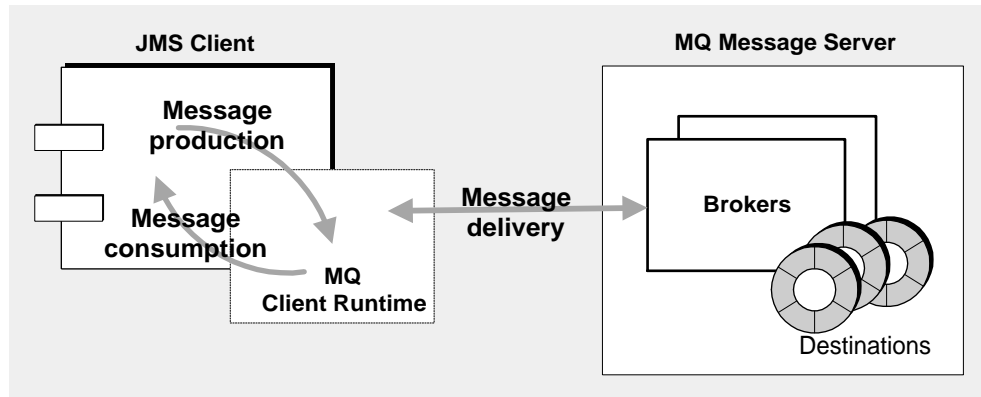
Queue Destinations. Queue destinations are used in point to point messaging, where a message is meant for ultimate delivery to only one of a number of consumers that has registered an interest in the destination. As messages arrive from producer clients, they are queued and delivered to a consumer client.

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.

The figure “[Messaging Operations](#)” on [page 291](#) 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 284](#)). 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 ONE Application Server 7, Enterprise Edition

MQ Platform Edition is automatically installed as part of the Sun ONE Application Server 7, Enterprise Edition installation process. For more information, see the *Sun ONE Application Server Installation Guide*.

This installation provides Sun ONE Application Server 7, Enterprise Edition with a JMS messaging system that supports any number of Sun ONE Application Server 7, Enterprise Edition instances. Each server instance, by default, has an associated built-in JMS Service that supports all JMS clients running in the instance.

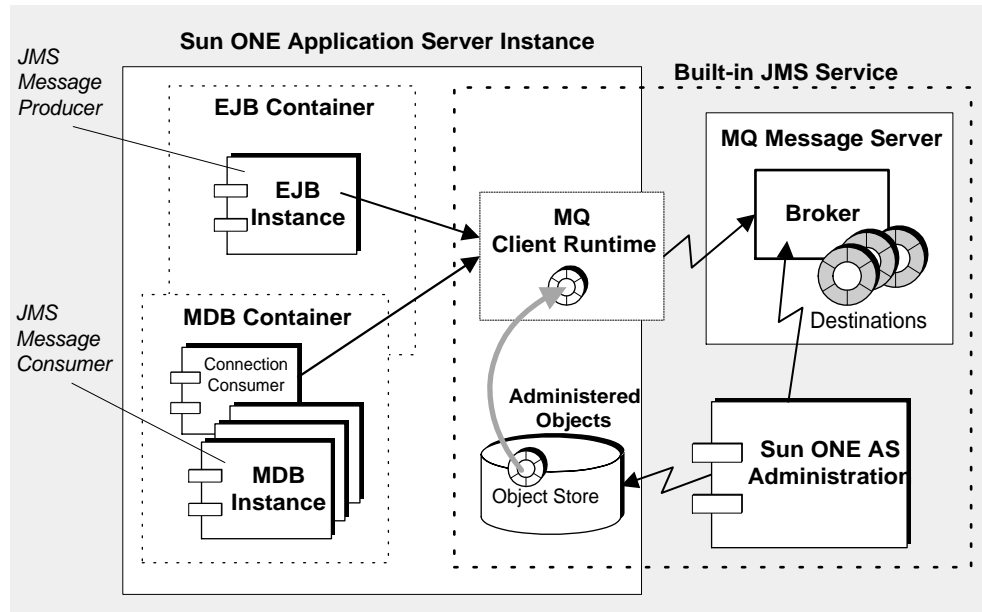
This topic covers the following topics:

- [Architecture of the Built-in JMS Service](#)
- [Disabling the Built-in JMS Service](#)

The built-in JMS Service can be administered using Sun ONE Application Server 7, Enterprise Edition administration tools (see [“Administration of the Built-in JMS Service”](#) on page 295).

Architecture of the Built-in JMS Service

The built-in JMS Service—illustrated in the figure [“Built-in MQ Messaging System”](#) on page 293—is similar to an ordinary MQ messaging system (shown in the figure [“MQ System Architecture”](#) on page 289), except for a number of qualifications, described below.

Figure 10-6 Built-in MQ Messaging System

MQ Message Server. Each Sun ONE Application Server 7, Enterprise Edition 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 Sun ONE Application Server 7, Enterprise Edition instance as shown in the figure “[Built-in MQ Messaging System](#).” 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 Sun ONE Application Server 7, Enterprise Edition configuration store (the `server.xml` file) and can be modified as described in “[Configuring the JMS Service](#)” on page 296.

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 Sun ONE Application Server 7, Enterprise Edition. 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 302, and they are accessed by JMS clients using JNDI lookup code.

Sun ONE Application Server 7, Enterprise Edition Administration. The Sun ONE Application Server 7, Enterprise Edition 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 Administrator's Guide*. For a comparison of MQ and Sun ONE Application Server 7, Enterprise Edition administration capability, see the table [“Comparison of Sun ONE Message Queue and Sun ONE Application Server 7, Enterprise Edition Administration Capability”](#) on page 296.

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 Sun ONE Application Server 7, Enterprise Edition 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 296.)

An external JMS Service is a messaging system that is not controlled from within Sun ONE Application Server 7, Enterprise Edition. 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 ONE 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 `imqBrokerHostName` and `imqBrokerHostPort` properties appropriately).
4. Use this connection factory resource when deploying JMS applications on the Sun ONE 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](#)
- [Administering the Built-in JMS Service Using the Command-Line Interface](#)

Administration can be performed using the Sun ONE Application Server 7, Enterprise Edition’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 ONE Message Queue and Sun ONE Application Server 7, Enterprise Edition Administration Capability.”](#)

Table 10-1 Comparison of Sun ONE Message Queue and Sun ONE Application Server 7, Enterprise Edition Administration Capability

Function	Sun ONE MQ Administration Tools	Sun ONE AS Administration interface	Sun ONE AS 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
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 ONE Application Server 7, Enterprise Edition 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 [“JMS Service Properties.”](#)

Table 10-2 JMS Service Properties

Property	Description	Default Value
Log level	The level of logging information you want written to the Sun ONE Application Server log file. For more information, see Chapter 5 , “Using Logging.”	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 Sun ONE Application Server 7, Enterprise Edition 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
Administrator's username/password	The username/password needed to perform broker administration tasks, such as managing physical destinations (see “Managing Physical Destinations” on page 299). If you want administrator access to a broker instance to be secure (by default, any user is provided access), you need to first make the appropriate entries in the broker's user repository (as described in the MQ Administrator's Guide) 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 MQ <i>Administrator's Guide</i> . (The <code>-name</code> and <code>-port</code> arguments, if supplied, are ignored.)	

Table 10-2 JMS Service Properties (*Continued*)

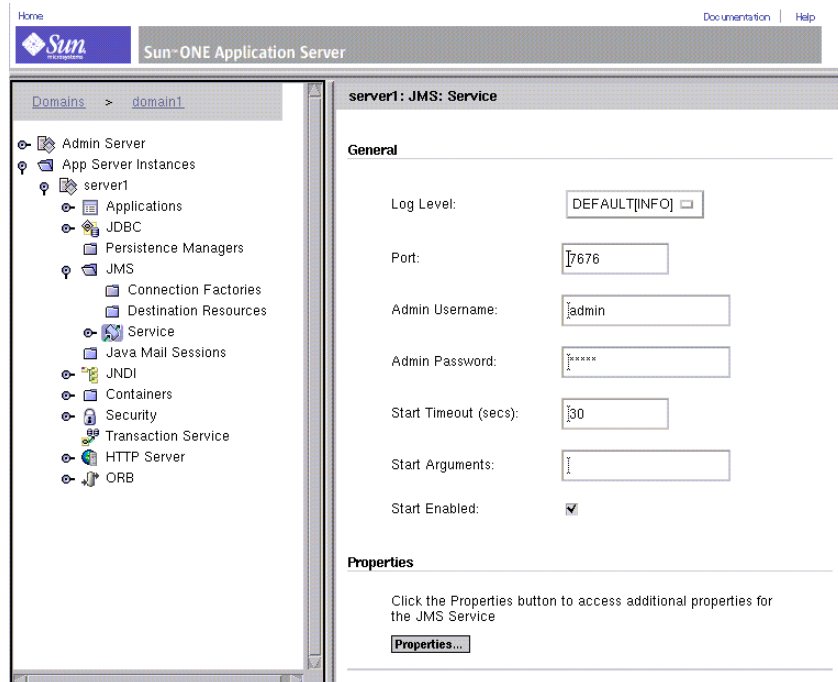
Property	Description	Default Value
Startup Enabled	Specifies whether the built-in JMS Service is started up when the server instance starts up. If you are not supporting JMS messaging or are using an external JMS message service, set this property to FALSE.	TRUE

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.

The JMS Service configuration screen is displayed on the right pane.

Figure 10-7 JMS Service Configuration Screen

5. Modify any of the properties you wish (see the table [“JMS Service Properties”](#) on page 297).
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 297).

Using the Administration interface, you can perform the following management tasks for physical destinations on a built-in JMS Service:

- [Create a Queue or Topic Destination](#)
- [List Physical Destinations](#)
- [Delete a Physical Destination](#)

Create a Queue or Topic Destination

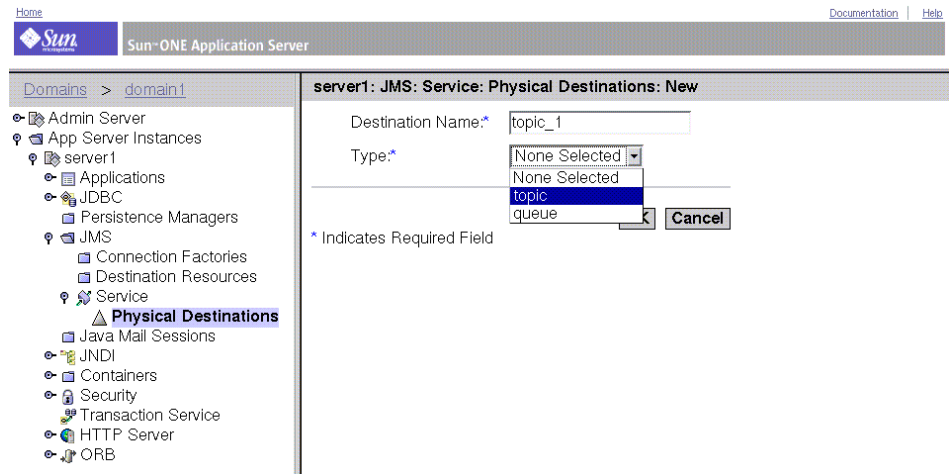
To create a queue or topic destination:

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.

Figure 10-8 New JMS Physical Destination Screen

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.

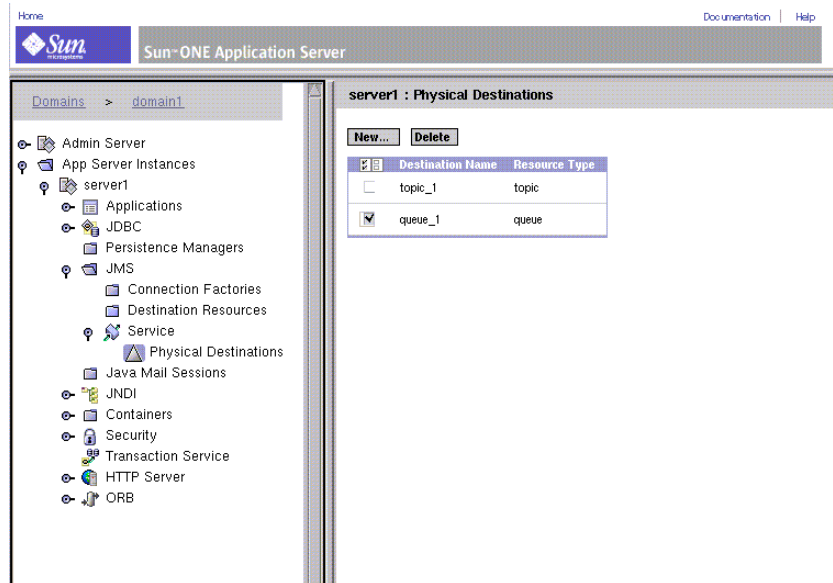
Delete a Physical Destination

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 301](#)).
2. Click within the Select box for each destination you wish to delete.

Figure 10-9 JMS Physical Destinations Screen



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 ONE Application Service 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 291](#)).

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 297](#)), 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

In the case of connection factory administered objects, the Administration interface creates, by default, a connection factory that points to the built-in JMS Service—that is, to a broker instance whose hostname is the local host and whose port number is that set when configuring the JMS Service (see the table [“JMS Service Properties” on page 297](#)).

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 and port number of the appropriate broker instance.

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:

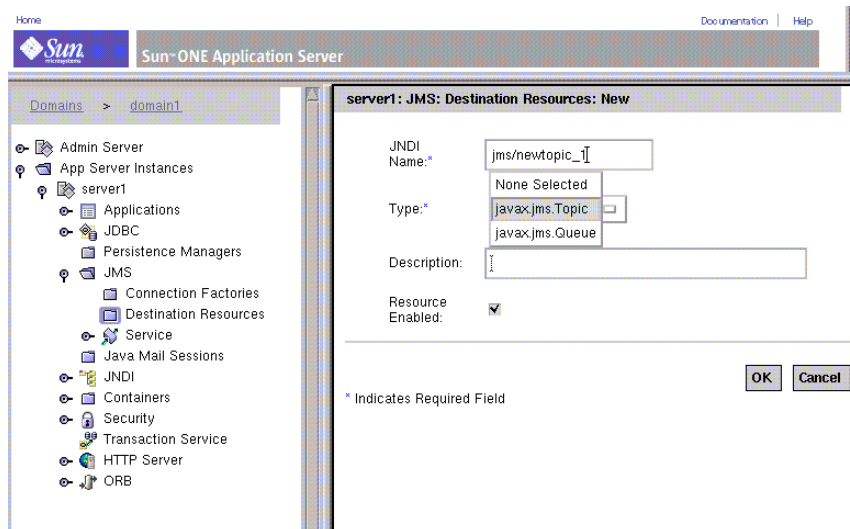
1. Open the Administration interface.
2. Open a server instance in the left pane.
3. Open the JMS folder.
4. Select the Destination Resources link.

The Destination Resources screen is displayed in the right pane.

5. Click on the New button.

The Destination Resources: New screen is displayed.

Figure 10-10 New Destination Administered Object Screen



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

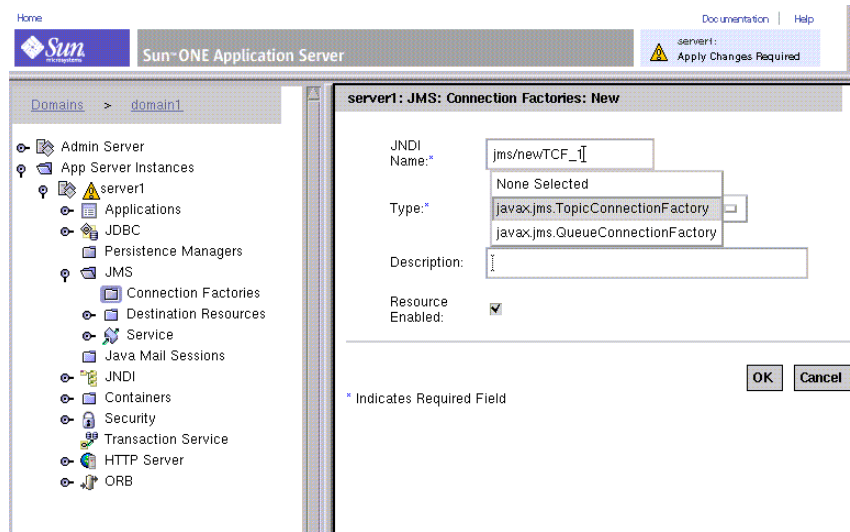
To create a queue connection factory or topic connection factory administered object:

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.

Figure 10-11 New ConnectionFactory Administered Object Screen

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 the `imqBrokerHostName` and `imqBrokerHostPort` 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 `imgBrokerHostName`.
8. In the Value field, enter a value for the property.
9. In the Name field and `imgBrokerHostPort`.
10. In the Value field, enter a value for the property.
11. 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.

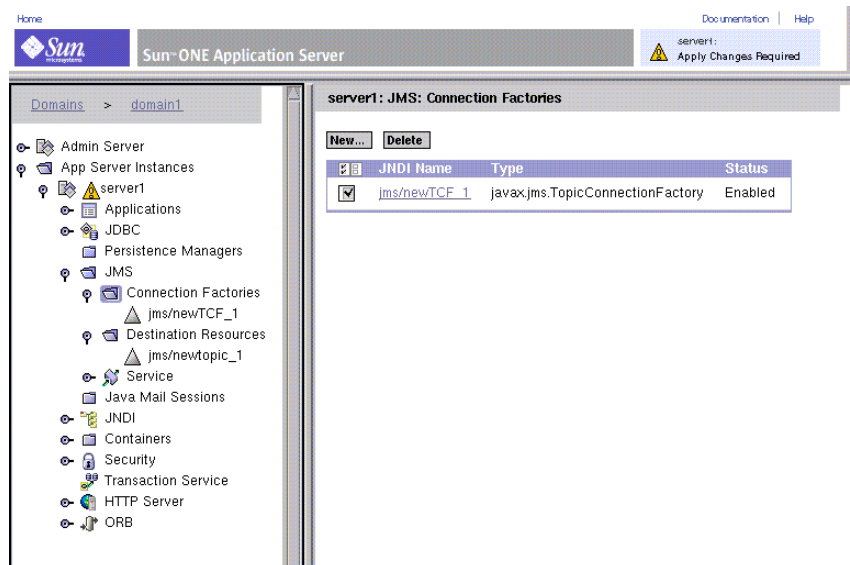
Delete an Administered Object Resource

To delete an administered object resource:

1. List the existing administered object resources (see [“List Administered Object Resources” on page 307](#)).

The right pane displays the existing administered object resources.

2. Click within the Select box for each object you wish to delete.

Figure 10-12 JMS Connection Factory Screen, Refreshed

3. Click the Delete button to remove each selected object.

The screen refreshes the list, showing the remaining administered object resources.

Administering the Built-in JMS Service Using the Command-Line Interface

The Sun ONE 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.

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.

asadmin Commands for Administering the Built-in JMS Service

Command	Use
create-jms-resource	Creates a JMS resource.
delete-jmsdest	Deletes a JMS physical destination.
delete-jms-resource	Deletes a JMS resource.
jms-ping	Pings the JMS provider to see if it is running.
list-jmsdest	Lists JMS physical destinations for a server instance.
list-jms-resources	Lists JMS resources for a server instance.
get and set jms-service	Gets and sets attributes for the JMS service.
get and set jms-resource	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 Sun ONE Application Server 7, Enterprise Edition 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, Sun ONE Application Server 7, Enterprise Edition 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.

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 Sun ONE Application Server 7, Enterprise Edition 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.

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 Sun ONE Application Server 7, Enterprise Edition. 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 Sun ONE Application Server 7, Enterprise Edition.

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 Sun ONE Application Server 7, Enterprise Edition instance for which you want to configure ORB settings.
2. Click the ORB tab. You will see the figure “General ORB Configuration” in the right pane of the Administration interface:

Figure 11-1 General ORB Configuration

General

Monitoring Enabled: ☒ ⓘ

Log Level: DEFAULT[INFO] ▼

Thread Pool

Steady Pool Size:

Max Pool Size:

Idle Timeout (secs):

Advanced

Max Message Fragment Size: 1024 ▼

Total Connections:

Save Reset

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 Sun ONE Application Server 7, Enterprise Edition 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 Sun ONE Application Server 7, Enterprise Edition 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 Sun ONE Application Server 7, Enterprise Edition 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 Sun ONE Application Server 7, Enterprise Edition.

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 Sun ONE Application Server 7, Enterprise Edition 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 Sun ONE Application Server 7, Enterprise Edition.
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 figure [Creating a New IIOP Listener](#):

Figure 11-2 Creating a New IIOP Listener

server1: ORB: IIOP Listeners: New

Id:*

Address:*

Port:

Listener Enabled: ☒

SSL/TLS Settings

Certificate Nickname:

SSL2 Enabled: ☐

SSL2 Ciphers: ☐ rc4 ☐ rc4export
☐ rc2 ☐ rc2export
☐ idea ☐ des
☐ desede3

SSL3 Enabled: ☐

TLS Enabled: ☒

TLS Rollback Enabled: ☒

SSL3/TLS Ciphers: ☒ rsa_rc4_128_md5 ☒ rsa_3des_sha
☒ rsa_des_sha ☐ rsa_rc4_40_md5
☐ rsa_rc2_40_md5 ☐ rsa_null_md5
☒ rsa_des_56_sha ☒ rsa_rc4_56_sha

Client Authentication Enabled: ☐

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 Sun ONE Application Server 7, Enterprise Edition. 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 Sun ONE Application Server 7, Enterprise Edition, 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 Sun ONE Application Server 7, Enterprise Edition is installed.
 - For more information about SSL settings for the listener path, and other details of security for Sun ONE Application Server 7, Enterprise Edition, see the *Sun ONE Application Server Administrator's Guide to Security*.
-

Deploying Applications

This chapter describes how to deploy various Sun ONE Application Server 7, Enterprise Edition modules and applications.

Sun ONE Application Server 7, Enterprise Edition modules and applications include J2EE standard elements and Sun ONE Application Server 7, Enterprise Edition specific elements. Only Sun ONE Application Server 7, Enterprise Edition specific elements are described in detail in this chapter.

To know about packaging and assembling modules and applications for deployment, see the *Sun ONE Application Server Developer's Guide*.

This chapter includes the following topics:

- [About J2EE Modules](#)
- [About J2EE Applications](#)
- [J2EE Standard Descriptors](#)
- [Sun ONE 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition. 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 ONE Application Server 7, Enterprise Edition connector has a J2EE XML file, `ra.xml`. A connector must also have a Sun ONE Application Server 7, Enterprise Edition 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, as in the following figure.

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 ONE Application Server 7, Enterprise Edition application `.ear` file ready for deployment.

Each module has a Sun ONE Application Server 7, Enterprise Edition deployment descriptor and a J2EE deployment descriptor. The Sun ONE Application Server 7, Enterprise Edition Administration interface uses the deployment descriptors to deploy the application components and to register the resources with the Sun ONE Application Server 7, Enterprise Edition.

An application consists of one or more modules, an optional Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server Developer's Guide*.

The following table, “[J2EE Standard Descriptors](#),” 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 12-1 J2EE Standard Descriptors

Deployment Descriptor	Where to Find More Information
<code>application.xml</code>	Java 2 Platform Enterprise Edition Specification, v1.3, Chapter 8, “Application Assembly and Deployment - J2EE:application XML DTD”
<code>web.xml</code>	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”
<code>ejb-jar.xml</code>	Enterprise JavaBeans Specification, v2.0, Chapter 16, “Deployment Descriptor”
<code>application-client.xml</code>	Java 2 Platform Enterprise Edition Specification, v1.3, Chapter 9, “Application Clients - J2EE:application-client XML DTD”
<code>ra.xml</code>	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/>

Sun ONE Application Server Descriptors

Sun ONE Application Server 7, Enterprise Edition uses additional deployment descriptors for configuring features specific to the Sun ONE Application Server 7, Enterprise Edition. 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 ONE Application Server Developer's Guide*.

The following table, “[Sun ONE Application Server 7, Enterprise Edition Descriptors](#),” shows where to find more information about Sun ONE Application Server 7, Enterprise Edition deployment descriptors. The left column lists the deployment descriptors, and the right column lists where to find more information about those descriptors.

Table 12-2 Sun ONE Application Server 7, Enterprise Edition Descriptors

Deployment Descriptor	Where to Find More Information
sun-application.xml	“The Application Deployment Descriptor Files” on page 340.
sun-web.xml	<i>Sun ONE Application Server 7, Enterprise Edition Developer’s Guide to Web Applications</i>
sun-ejb-jar.xml and sun-cmp-mapping.xml	<i>Sun ONE Application Server 7, Enterprise Edition Developer’s Guide to Enterprise Java Beans</i>
sun-application-client.xml and sun-acc.xml	<i>Sun ONE Application Server 7, Enterprise Edition Developer’s Guide to Clients</i>
sun-ra.xml	<i>Sun ONE J2EE CA Service Provider Implementation Administrator’s Guide</i>

NOTE	The Sun ONE Application Server 7, Enterprise Edition deployment descriptors must have 600 level access privileges on UNIX systems.
-------------	--

The DTD schema files for all the Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition. 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition, 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

```

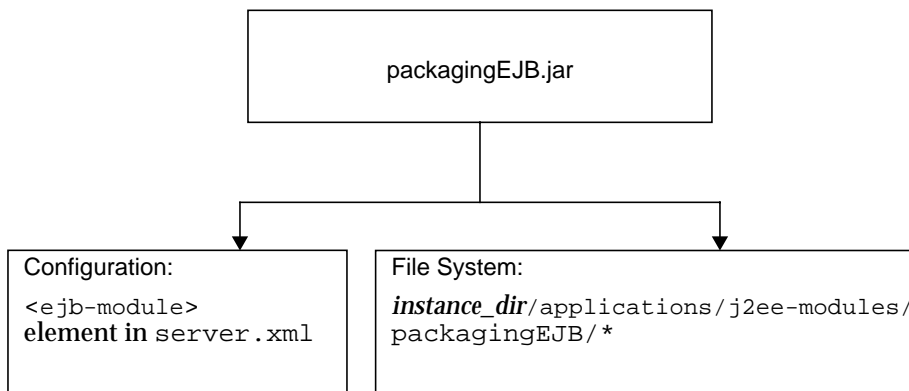
Runtime Environments

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 12-1 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>

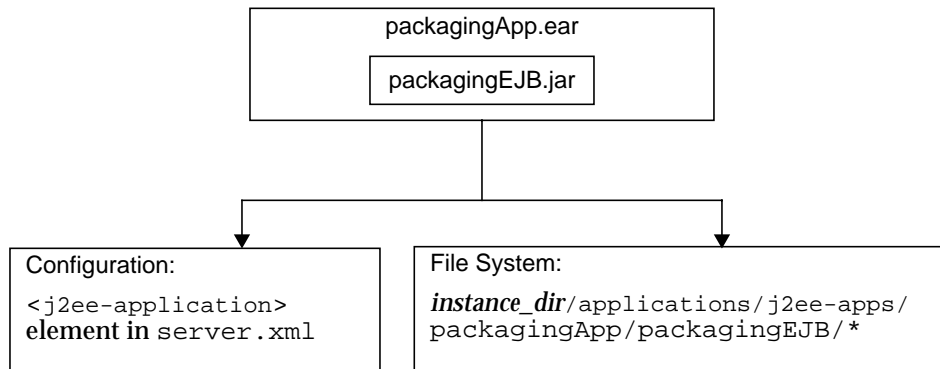
```

The *type* of the module in `server.xml` can be `lifecycle`, `ejb`, `web`, or `connector`. For details about `server.xml`, see the *Sun ONE Application Server 7, Enterprise Edition Administrator's Configuration File Reference*.

Application Runtime Environment

The following figure, “[Application Runtime Environment](#),” illustrates the environment for application-based deployment.

Figure 12-2 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 ONE Application Server 7, Enterprise Edition Administrator's Configuration File Reference*.

About Classloaders

Understanding Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server Developer's Guide*.

Deploying Modules and Applications

This section describes the different ways to deploy J2EE applications and modules to the Sun ONE Application Server 7, Enterprise Edition. It covers the following topics:

- [Deployment Names and Errors](#)
- [The Deployment Life Cycle](#)
- [Deployment of 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 325](#).

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 ONE Application Server 7, Enterprise Edition 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.

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 ONE Application Server 7, Enterprise Edition 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:
 - a. Open the Applications component under your server instance.
 - b. Go to the Applications page.
 - c. Check the Reload Enabled box to enable dynamic reloading.
 - d. 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.
 - e. Click on the Save button.
 - f. Go to the server instance page and select the Apply Changes button.
- 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 ONE Application Server 7, Enterprise Edition 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"`.

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 337](#).

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] [--virtualservers
virtual_servers] [--type application|ejb|web|connector] [--contextroot
contextroot] [--force=true] [--precompilejsp=false] [--name
component_name] [--upload=true] [--retrieve local_dirpath] [--instance
instance_name] filepath
```

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 326](#). 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] [--virtualservers
virtual_servers] [--type application|ejb|web|connector] [--contextroot
contextroot] [--force=true] [--precompilejsp=false] [--name
component_name] [--instance instance_name] dirpath
```

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] [--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 ONE Application Server 7, Enterprise Edition 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. Click on the OK button.

To deploy a lifecycle module, see [“Deploying a Lifecycle Module” on page 337](#).

Sun ONE Studio

You can use Sun ONE Studio 4 to deploy J2EE applications and modules. For more information about using Sun ONE Studio, see the *Sun ONE Studio 4, Enterprise Edition Tutorial*.

NOTE	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.
-------------	---

Deployment of 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 328](#).

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 333](#).

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 ONE Application Server 7, Enterprise Edition 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 333](#).

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 ONE Application Server 7, Enterprise Edition Administrator's Configuration File Reference*.

Deploying a Lifecycle Module

For general information about lifecycle modules, see the *Sun ONE 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]
[--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]
[--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] 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 on the OK button.

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 ONE Application Server 7, Enterprise Edition 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 333](#).

Deploying Static Content

Static content (HTML, images, etc.) can be hosted both on the web server and on the Sun ONE Application Server 7, Enterprise Edition. However, when a WAR is registered, the static content gets deployed on the application server. All of the samples shipped with Sun ONE Application Server 7, Enterprise Edition 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/context_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 330](#).

The Application Deployment Descriptor Files

Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition specific file (`sun-application.xml`), described in this section.

For more information on application deployment descriptor files, see the *Sun ONE Application Server Developer's Guide*.

Managing HTTP Server Features and Virtual Servers

Chapter 13, “Configuring HTTP Features”

Chapter 14, “Using Virtual Servers”

Chapter 15, “Managing Virtual Server Content”

Configuring HTTP Features

This chapter describes how to configure preferences for your HTTP-related features of your Sun ONE Application Server. For preferences related to virtual servers and HTTP listeners, see [Chapter 14, “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](#)
- [Configuring the File CacheEditing Advanced Settings](#)
- [Configuring MIME Types](#)

About the HTTP Features

The Sun ONE 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 348](#).

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 ONE Application Server Administrator’s Configuration File Reference*.

Configuring the File Cache

The Sun ONE 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 ONE 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 ONE 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 ONE 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 ONE 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) or 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] [--bwlmit bandwidth_limit]
[--enforcebwlmit enforce_bandwidth_limit] [--connlmit connection_limit]
[--enforceconnlmit 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 151.](#)

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 plugins. 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 ONE 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 ONE 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 ONE 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 14, “Using Virtual Servers.”](#)

Using Virtual Servers

This chapter explains how to set up and administer virtual servers using your Sun ONE Application Server. For information on configuring settings for virtual server content, see [Chapter 15, “Managing Virtual Server Content.”](#)

This chapter includes the following topics:

- [Virtual Servers Overview](#)
- [Using Sun ONE 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 ONE 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 362](#). For more information on deploying virtual servers, see [“Deploying Virtual Servers.”](#)

This virtual server controls the Sun ONE 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. For more information about this file, see the *Sun ONE 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](#)
- [The 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 359](#).

When you install the unbundled version of Sun ONE 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 ONE 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 356](#).

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 `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 369](#).

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 ONE 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.

NOTE	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.

The 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 ONE 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 ONE 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 ONE 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 ONE 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 ONE 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 ONE Application Server Features with Virtual Servers

Sun ONE 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 ONE 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 ONE 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 ONE Application Server and virtual servers, see the *Sun ONE 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 370](#).

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 ONE 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 ONE 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 356](#).

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 349](#).

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 ONE 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 356](#).

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 ONE Application Server 7 supports the Servlet 2.3 API specification, which allows servlets and JSPs to be included in web applications. In addition, Sun ONE 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 151](#).

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 362](#). Also see the online help.

Editing General Settings Using the Command-Line Interface

You can also edit these settings 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 ONE 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 15, “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 ONE 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 ONE Application Server.

- [Example 1: Default Configuration](#)
- [Example 2: Secure Server](#)
- [Example 3: Intranet Hosting](#)
- [Example 4: Mass Hosting](#)

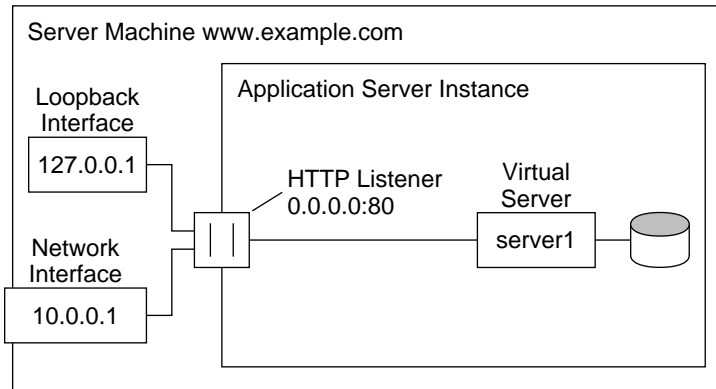
Example 1: Default Configuration

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 14-1 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/`

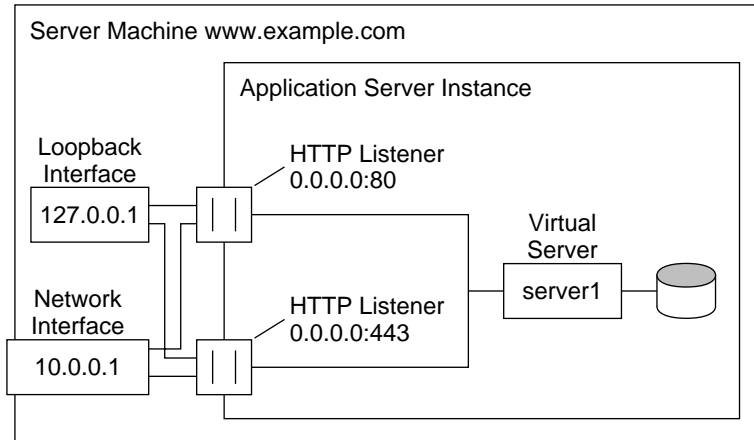
Use this configuration for traditional HTTP server use. You do not need to add additional virtual servers or HTTP listeners. You configure the settings of the server by changing the settings for server1.

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 14-2 Secure Server



DNS

<code>www.example.com</code>	<code>10.0.0.1</code>

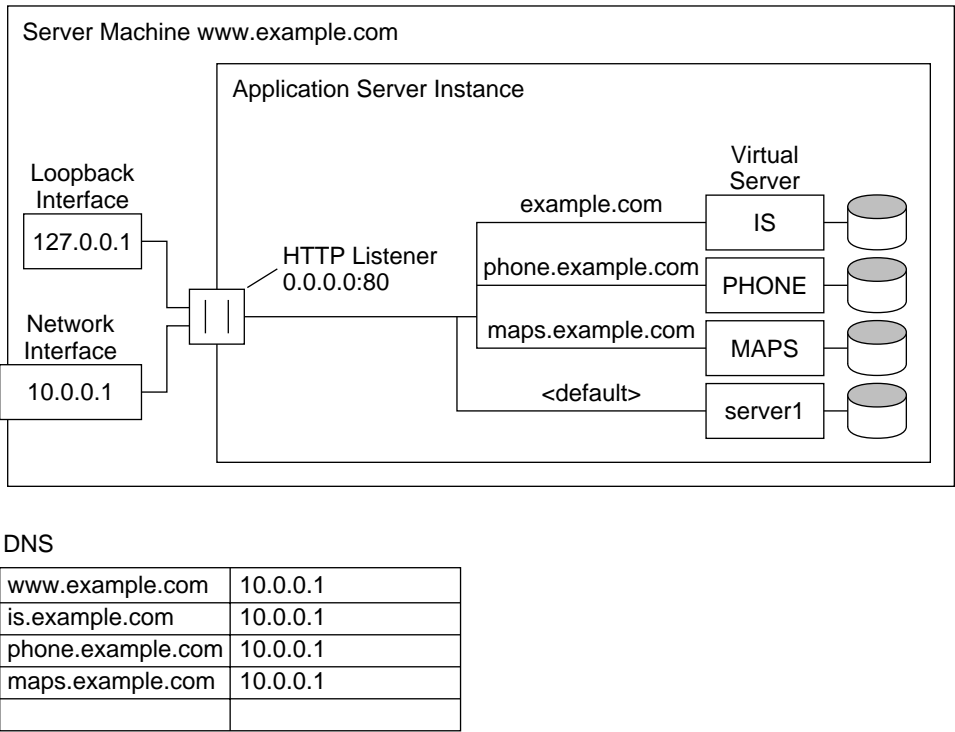
Please note that the SSL parameters are attached to the HTTP listener.

Example 3: Intranet Hosting

A more complex configuration of the Sun ONE 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 14-3 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 ONE 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:

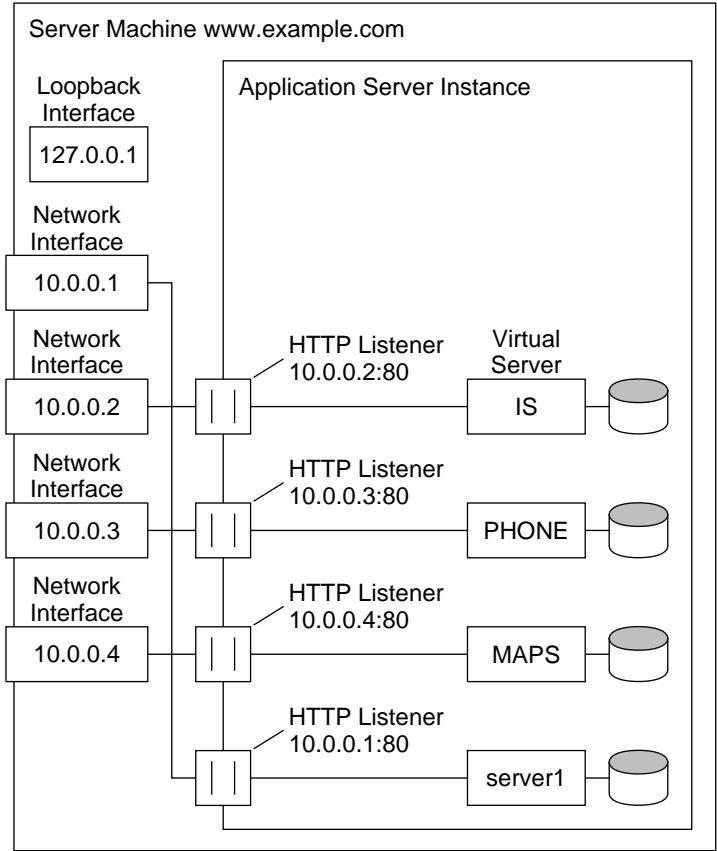
- They work with older clients that do not support the HTTP/1.1 Host header.
- Providing SSL support is straightforward.

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 14-4 Intranet Hosting Using IP-Address-Based Virtual Servers

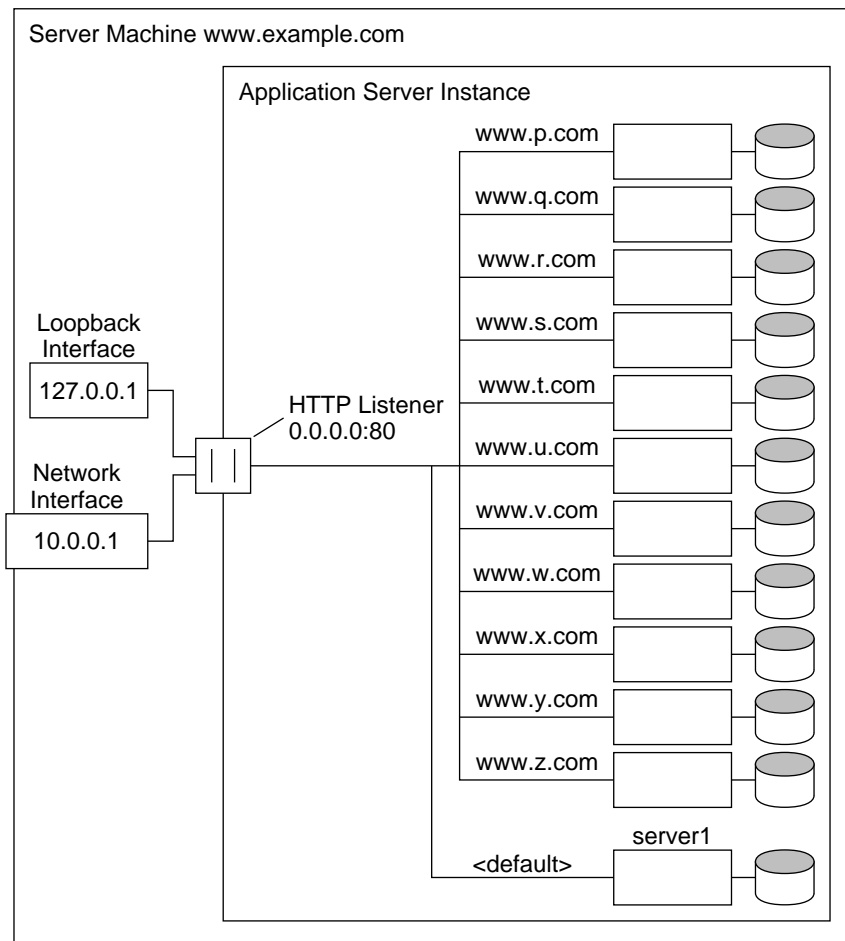


DNS

www.example.com	10.0.0.1
is.example.com	10.0.0.2
phone.example.com	10.0.0.3
maps.example.com	10.0.0.4

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 14-5 Mass Hosting**DNS**

<code>www.example.com</code>	<code>10.0.0.1</code>
<code>www.p.com</code>	<code>10.0.0.1</code>
<code>www.q.com</code>	<code>10.0.0.1</code>
<code>www.r.com</code>	<code>10.0.0.1</code>
...	
<code>www.z.com</code>	<code>10.0.0.1</code>

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 356](#).

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 Sun ONE Application Server 7, Enterprise Edition standard access controls (standard access controls are always applied before any `htaccess` access controls).

For information on using `htaccess`, see the *Sun ONE 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 383](#).

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 ONE 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 384](#). 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-`):

- | | |
|----------------------------|---------------------------|
| • <code>us-ascii</code> | • <code>iso-8859-1</code> |
| • <code>iso-2022-jp</code> | • <code>x-sjis</code> |

- `x-euc-jp`
- `x-mac-roman`

Additionally, the following aliases are recognized for `us-ascii`:

- `ansi_x3.4-1968`
- `ansi_x3.4-1986`
- `ascii`
- `us`
- `cp367`
- `iso-ir-6`
- `iso_646.irv:1991`
- `iso646-us`
- `ibm367`

The following aliases are recognized for `iso_8859-1`:

- `latin1`
- `iso_8859-1:1987`
- `ibm819`
- `iso_8859-1`
- `iso-ir-100`
- `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 ONE Application Server Developer's Guide to Web Applications*.

Setting Cache Control Directives

Cache-control directives are a way for Sun ONE 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 ONE Application Server Administrator's Guide to Security*.

Configuring Multiple Server Instances

Chapter 16, “Configuring Load Balancing”

Chapter 17, “Managing Clusters”

Chapter 18, “Configuring Session Persistence”

Chapter 19, “Using the cladmin Command”

Chapter 20, “Administering the High-Availability Database”

Configuring Load Balancing

This chapter describes how to setup load balancing of HTTP requests, configure, and administer the load balancer plug-in.

This chapter includes the following topics:

- [About Load Balancing](#)
- [Configuring Load Balancer Plug-in](#)
- [About Dynamic Reconfiguration](#)
- [Monitoring the Load Balancer Plug-in](#)
- [Load Balancer Configuration File](#)
- [Known Issues in Load Balancing Requests](#)

About Load Balancing

Sun™ Open Net Environment (ONE) Application Server 7, Enterprise Edition, provides load balancing and HTTP session persistence. The goal of load balancing is to evenly distribute the workload between multiple Sun ONE Application Server instances, thereby increasing the overall throughput of the system.

The load balancer plug-in is an enhanced version of the HTTP reverse proxy plug-in used in Sun ONE Application Server 7, Standard Edition. For more information on the reverse proxy plug-in, see Chapter 7, “Configuring the Web Server Plugin,” in *Sun ONE Application Server 7, Standard Edition Administrator's Guide*.

This section addresses the following topics:

- [Load Balancing Algorithm](#)

- [Requirements for Load Balancing](#)

Load Balancing Algorithm

Sun ONE Application Server load balancer uses 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 plugin 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.

NOTE	The browser 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 browser does not support cookies.

Requirements for Load Balancing

You must meet the following requirements before you can use the load balancer plug-in:

- Sun ONE Application Server 7, Enterprise Edition is installed.
- A web server is installed (Sun ONE or Apache).
- At least two application server instances are created and configured.
- Deployed your web applications to all application server instances that will participate in load balancing.

Configuring Load Balancer Plug-in

This section discusses the following topics:

- [Configuration Changes to Web Server](#)
- [Creating the Load Balancer Configuration File](#)
- [Configuring Multiple Web Server Instances](#)
- [Configuring Response Timeout](#)
- [Configuring HTTPS Routing](#)
- [Configuring Log Messages](#)
- [Configuring the Health-Checker](#)
- [Configuring Quiescence](#)

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 ONE Web Server Configuration](#)
- [Modifications to Apache Web Server](#)

NOTE The load balancer plug-in can be installed either along with Sun ONE Application Server 7, Enterprise Edition, or separately, on a machine running the supported web server.

For complete details on the installation procedure, see *Sun ONE Application Server Installation Guide*.

Modifications to Sun ONE Web Server Configuration

The installation program makes the following changes to the Sun ONE 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/webserver_instance/plugins/lbplugin
/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrou
gh" 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 the Load Balancer Configuration File” on page 401](#).

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.”](#)

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/config/loadbalancer.xml
locale en
</IfModule>

<VirtualHost machine_ip_address>
DocumentRoot "webserver_instance/apache/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.
- We recommend that the URL pattern be set to / to ensure that all requests are forwarded to the load balancer plug-in first. 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 Application Server 7 EE, the load balancing plug-in 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 the Load Balancer Configuration File

The `loadbalancer.xml` file should be created based on the `sun-loadbalancer_1_0.dtd` and `loadbalancer.xml.example` files installed during load balancer installation. These files are available in the following location:

- On Sun ONE 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 [“Load Balancer Configuration File” on page 413](#).

Sample loadbalancer.xml file

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun
ONE Web Server 6.0//EN"
"file:///webserver_install_dir/webserver_instance/config/sun-loa
dbalancer_1_0.dtd">

<loadbalancer>

<cluster name="cluster1">

<instance name="server1" enabled="true"
listeners="http://austen.sun.com:80
https://http://austen.sun.com:443"/>

    <instance name="server2" enabled="true"
listeners="http://polar.sun.com:80 https://polar.sun.com:443"/>

        <web-module context-root="webapps-simple" enabled="true"/>

</cluster>

<property name="response-timeout-in-seconds" value="120"/>
<property name="reload-poll-interval-in-seconds" value="180"/>
<property name="https-routing" 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`.
- The `listeners` 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 its 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 its port number, 443.

- 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_0.dtd">`
- For more information on creating and configuring application server instances, see [Chapter 4, “Using 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 17, “Managing Clusters.”](#)

For more information on `loadbalancer.xml`, see [“Load Balancer Configuration File” on page 413](#).

Configuring Multiple Web Server Instances

The Sun ONE Application Server 7, Enterprise Edition, 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 ONE Web Server Instances](#)
- [Configuring Multiple Apache Web Server Instances](#)

Configuring Multiple Sun ONE 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/webserver_instance/plugins/lbplugin
/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrou
gh" 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_0.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 the Load Balancer Configuration File” on page 401](#).

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/config/loadbalancer.xml
locale en
</IfModule>

<VirtualHost machine_ip_address>
DocumentRoot "webserver_instance/apache/htdocs"
ServerName server_name
</VirtualHost>

##END EE LB Plugin Parameters
```

2. Copy the `sun-loadbalancer_1_0.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 the Load Balancer Configuration File” on page 401](#).

Configuring Response Timeout

The load balancer configuration file, `loadbalancer.xml`, contains a `response-timeout-in-seconds` property that 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 HTTPS Routing

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 ONE 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, and there is no HTTPS listener healthy in the cluster, then the request is failed.
-

Configuring the Health-Checker

The load balancer periodically checks all the configured Sun ONE 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 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 17, “Managing Clusters.”](#)

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.

About 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 has been made to the `loadbalancer.xml` file, then the load balancer automatically reconfigures itself.

The time interval after which the load balancer checks the time stamp of the `loadbalancer.xml` file can be specified in the `reload-poll-interval-in-seconds` property in the `loadbalancer.xml` file. By default, this value is set to 0 seconds, and therefore, dynamic reconfiguration is not enabled.

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.

The Sun ONE Web Server's error log file can be found at this location:
webserver_install_dir/webserver_instance/logs/.

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 logging mechanism to write log messages. The load balancer plug-in uses the default logging level on Sun ONE 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.

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`.
For Apache Web Server, set the log level to `DEBUG`.
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
```

Load Balancer Configuration File

The `loadbalancer.xml` file contains the configuration information for load balancing Sun™ Open Net Environment (ONE) Application Server instances. This file is part of the load balancer plugin 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 ONE 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_0.dtd`, determines the format and content of the `loadbalancer.xml` file.

This chapter describes `loadbalancer.xml` and `sun-loadbalancer_1_0.dtd` files in these sections:

- [The `sun-loadbalancer_1_0.dtd` File](#)
- [Elements in the `loadbalancer.xml` File](#)

The `sun-loadbalancer_1_0.dtd` File

The `sun-loadbalancer_1_0.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.

NOTE Do not edit the `sun-loadbalancer_1_0.dtd` file; its contents change only with new versions of Sun ONE Application Server Enterprise Edition.

NOTE The `sun-loadbalancer_1_0.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 16-1 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 ONE 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.

NOTE	Subelements must be defined in the order in which they are listed under each Subelements 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 16-2 `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 16-3 `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 16-4 `property` attributes

Attribute	Default	Description
<code>name</code>	none	Specifies the name of the property.
<code>value</code>	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 16-5 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 16-6 `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 16-7 `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 16-8 `instance` attributes

Attribute	Default	Description
<code>name</code>	<code>none</code>	Specifies the name of the server instance. Within a cluster, instance names must be unique.
<code>enabled</code>	<code>true</code>	(optional) Specifies whether instance is active (enabled) for requests to be load balanced to it.

Table 16-8 instance attributes

Attribute	Default	Description
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, listener URLs must be unique.

web-module

Defines a web module.

Subelements

none

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 16-9 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.

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 16-10 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.

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`.

Managing Clusters

The use of clusters helps you to achieve high availability for your applications. This chapter explains how to use the load balancer to manage the clusters. It also provides important information about clusters that will help you fully utilize the failover capabilities of SunTM Open Net Environment (Sun ONE) Application Server 7, Enterprise Edition.

This chapter includes the following sections:

- [About Clusters](#)
- [Understanding Assigned Requests and Unassigned Requests](#)
- [Defining a Cluster](#)
- [Adding Application Server Instances to a Cluster](#)
- [Setting up Multiple Clusters](#)
- [Deploying a Web Application on a Cluster](#)
- [Sample Cluster Configuration in a loadbalancer.xml File](#)
- [Starting a Cluster](#)
- [Understanding Quiescing](#)
- [Using Multiple Clusters for Online Upgrades Without Loss of Service](#)
- [Reconfiguring an Application Server Instance in a Running Cluster](#)
- [Undeploying a Web Application from a Cluster](#)
- [Redeploying an Existing Web Application on a Cluster](#)
- [Stopping an Application Server Instance in a Cluster](#)
- [Stopping a Cluster](#)

- [Removing an Application Server Instance from a Cluster](#)
- [Removing a Cluster](#)
- [Using Multiple Load Balancers](#)

About 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 ONE Application Server helps you to achieve:

- High availability by allowing for failover protection for the 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 ONE Application Server distributes requests to the available application server instances within the cluster. There is no disruption in service required as you add more application server instances to a cluster.

NOTE You can use multiple clusters to perform online upgrades without loss of service. This is explained in [“Using Multiple Clusters for Online Upgrades Without Loss of Service”](#) on page 439.

Important Facts About Clusters in Sun ONE Application Server

Here are important facts that you must know about clusters in Sun ONE Application Server:

- Sun ONE Application Server supports failover for HTTP 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 18, “Configuring Session Persistence.”](#)

- The application server instances in a Sun ONE 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.

For these applications to take part in failover, they must be *distributable*, that is, enabled for session persistence.

For more information, see [Chapter 18, “Configuring Session Persistence.”](#)

- All the instances should point to the same session store. For more information, see [“Creating a Session Store” on page 451.](#)
- The value of the `cluster-id` property should be the same for all application server instances. For more information, see [“Specifying a Cluster Name for Application Server Instance” on page 482.](#)
- Each application server instance in a cluster can host more than one application.
- Although you can add the same application server instance to multiple clusters, it is not recommended because of the resulting management complexity. For example, this might create problems while an instance that is part of more than one cluster is removed from 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.
- The application server instances within a cluster can be hosted on different machines or on the same machine. That is, you can group application server instances across different machines into a cluster.
- A particular load balancer can forward requests to application server instances on multiple clusters. You can use this ability of the load balancer 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 439.](#)
- 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` file must be identical.

- Each session is tied to a particular cluster. What this implies is that 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. This lets you use the load balancer and upgrade components within the Sun ONE Application Server without loss of service. For more information, see [“Using Multiple Clusters for Online Upgrades Without Loss of Service” on page 439](#).

Understanding Assigned Requests and Unassigned Requests

When a request first comes in from a 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 [“Load Balancing Algorithm” on page 396](#).

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* request.

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 `time-out-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.

Example: Defining a Cluster

```
<cluster name=mycluster1>

<health-checker url="/" interval-in-seconds="10"
time-out-in-seconds="20"/>

</cluster>
```

About 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.

The following table describes the attributes of `instance`. The leftmost column specifies the name of the attribute, the next column provides the default value of the attribute, the next column specifies whether it is mandatory to specify the attribute and the rightmost column provides a description of the attribute.

Table 17-1 Attributes of instance Element

Name of Attribute	Default Value	Required (Yes or No)	Description
name	No default value specified	Yes	Specifies the name of the application server instance in the <code>loadbalancer.xml</code> file. This name need not be the same as the name of the application server instance that you specified when creating the instance. The name must be unique for a cluster, that is, no two application server instances within a cluster can have the same name.
enabled	true	No	Specifies whether the application server instance is enabled or disabled.
disable-timeout-in-minutes	31	No	Specifies the quiescing period in minutes for the application server instance. For more information, see “Understanding Quiescing” on page 434 .
listeners	No default value specified	Yes	Specifies the HTTP listeners for the application server instance. You can specify multiple listeners for an application server instance by separating the URLs of the listeners with a space.

Difference Between Enabling and Starting an Application Server Instance

Enabling an application server instance is different from starting the application server instance. Even if you start an application server instance and define it to be a part of the cluster, the application server instance will not receive requests from the load balancer until you enable it. Enabling an application server instance that is part of a cluster means that the load balancer will route requests to this application server instance. Enabling, therefore, makes the application server instance an active part of the cluster. Similarly, disabling an application server instance is different from stopping the application server instance.

Sometimes you may need to make configuration changes to an application server instance that require a restart of the application server instance. In such cases, you must follow the correct sequence of disabling the application server instance, stopping it, starting it, and enabling it. For more information on such scenarios, see [“Reconfiguring an Application Server Instance in a Running Cluster” on page 440](#).

NOTE After you mark an application server instance in the `loadbalancer.xml` file as enabled or disabled, the instance is enabled or disabled when the load balancer next polls the `loadbalancer.xml` file. For more information, see [“About Dynamic Reconfiguration” on page 409](#).

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 79](#).
- Any new application server instance that you add to a cluster must have the same version and the same patch level of Sun ONE 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 ONE 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 [“Creating a Session Store” on page 451](#).
- The value of the `cluster-id` property should be the same for all application server instances. For more information, see [“Specifying a Cluster Name for Application Server Instance” on page 482](#).
- The clocks of all the computers on which application server instances in a cluster reside must be synchronized.

Adding Application Server Instances to a Cluster

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. The name must be unique for a cluster, 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. For more information on enabled and disabled application server instance, see [“Difference Between Enabling and Starting an Application Server Instance” on page 429](#).
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 disable the application server instance, this is the time interval after which the load balancer stops sending assigned requests to the application server instance. For more information on quiescing and quiescing period, see [“Understanding Quiescing” on page 434](#).

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>
```

Setting up 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 426](#) and [“Adding Application Server Instances to a Cluster” on page 430](#).

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 [“Creating a Session Store” on page 451](#).

Deploying a Web Application on a Cluster

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 [Chapter 19, “Using the cladmin Command.”](#) For more information on deploying an application to an application server instance, see [“Tools for Deployment” on page 333.](#)

2. In the `loadbalancer.xml` file, create a new `web-module` sub-element for 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 [“Understanding Quiescing” on page 434](#).

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>
```

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 [“Load Balancer Configuration File” on page 413](#).

```
<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>
```

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 [Chapter 19, “Using the cladmin Command.”](#)

2. Enable all the application server instances in the cluster.

Understanding Quiescing

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 430](#).

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.

NOTE	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 “Making an Application Distributable” on page 483 .
-------------	---

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 [“About Dynamic Reconfiguration” on page 409](#). 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 [“About Dynamic Reconfiguration” on page 409](#).

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 ONE Application Server without any loss of service. A component can, for example, be a JVM, the Sun ONE 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 442](#).
2. Upgrade the component in that cluster.
3. Start the cluster. For more information on starting a cluster, see [“Starting a Cluster” on page 434](#).
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 20, “Administering the High-Availability Database.”](#)
 - 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. Disable the application server instance from the cluster 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, reconfigure the application server instance.
3. Restart the application server instance.
4. Enable the application server instance by setting the value (in the `loadbalancer.xml` file) of the `enabled` attribute of the `instance` element to `true`.

Undeploying a Web Application from a Cluster

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

For more information on the `cladmin` command, see [Chapter 19, “Using the `cladmin` Command.”](#) For more information about undeploying an application from an application server instance, see [“Tools for Deployment” on page 333.](#)

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 ONE 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.

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 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.

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 [“Undeploying a Web Application from a Cluster” on page 440](#).
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.

Using Multiple Load Balancers

For a set of clusters, you can use multiple load balancers to improve scalability by adding more application server instances and to improve fault tolerance.

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 operational 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

This chapter describes how to configure session persistence for SunTM Open Net Environment (Sun ONE) Application Server 7, Enterprise Edition.

This chapter includes the following sections:

- [About Session Persistence](#)
- [Creating a Session Store](#)
- [Configuring JDBC Parameters for the HADB Database](#)
- [Enabling Availability for an Application Server Instance](#)
- [Disabling Availability for an Application Server Instance](#)
- [Configuring Session Persistence Options](#)
- [Specifying JNDI Name of the JDBC Resource for HADB Database](#)
- [Specifying a Cluster Name for Application Server Instance](#)
- [Making an Application Distributable](#)
- [Default Values for Session Persistence](#)
- [Clearing the Session Store](#)

About Session Persistence

As an application session proceeds, there is often data that is part of the session that is not stored in a traditional database. An example of such data is the content of your shopping cart. Sun ONE 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.

Sun ONE Application Server supports persistence of HTTP sessions. Failover of certain J2EE object references that are stored in HTTP sessions is also supported.

The high-availability database (HADB) bundled with the Sun ONE Application Server works as the persistence store to provide high availability for applications.

NOTE [Appendix B, “Failover Scenarios”](#) describes typical failover scenarios in detail.

This section includes the following topics:

- [Understanding Persistence Types](#)
- [Configuring Session Persistence for Application Server Instances and Applications](#)
- [Failover of J2EE Object References Stored in HTTP Sessions](#)
- [Basic Session Persistence Configuration Steps](#)

Understanding Persistence Types

The *persistence type* defines if and how HTTP session persistence and failover take place in the Sun ONE Application Server. The persistence types available 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 HTTP session persistence and failover capabilities.

The `memory` persistence type provides no failover and is the default persistence type. It is the best choice for production environments that *do not* require HTTP session persistence and failover capabilities.

In addition, the `memory` and the `file` persistence types are useful for development environments where a developer may want to test session persistence and failover without having to install and set up HADB.

The persistence types are described in [“Setting the Persistence Type” on page 462](#).

Configuring Session Persistence for Application Server Instances and Applications

You can configure session persistence for application server instances in a cluster through the Administration interface or through the command-line interface. These settings will apply to all applications that are deployed on the application server instance.

NOTE For changes in session persistence configuration to take effect, you must manually apply the configuration and restart the application server instance.

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 [“Understanding Quiescing” on page 434](#) and [“Stopping an Application Server Instance in a Cluster” on page 441](#).

For any application, you can specify the session persistence settings through the corresponding `sun-web.xml` file.

NOTE Application-level session persistence configuration takes precedence over the instance-level session persistence configuration.

Even if you change the instance-level configuration *after* an application has been deployed, the settings for the application still override the settings for the application server instance.

You may want to specify the session persistence settings for an 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 469](#).

NOTE	Session state will not be 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 “Enabling Availability for an Application Server Instance” on page 456 . For more information on setting the persistence type, see “Setting the Persistence Type” on page 462 .
-------------	---

NOTE	If you make session persistence configuration changes for an application, you must redeploy the application before the changes can take effect.
-------------	---

Failover of J2EE Object References Stored in HTTP Sessions

Sun ONE Application Server supports failover of J2EE object references that are stored in HTTP Sessions. The Table “Object Types Supported for J2EE Web Application Session State Failover” on page 449 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 RMI or IIOP 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 (see the table [Table 18-1 on page 449](#)).

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.

- If an application references a J2EE component or resource that is located on a different server instance, the reference may be lost.
- References to open files or network connections are lost.

For information about how to work around some of these limitations, see *Sun ONE Application Server Application Design Guidelines for Storing Session State*.

In the following table, the left column specifies the Java Object Type and the right column specifies whether failover support is provided—Yes in the Failover support column means that failover support is provided and No in the Failover Support column means that failover support is not provided.

Table 18-1 Object Types Supported for J2EE Web Application Session State Failover

Java Object Type	Failover Support Provided
EntityBean local home reference, local object reference	Yes
Stateful SessionBean local home reference	Yes
Stateful SessionBean local object reference	No
Stateless SessionBean local home reference, local object reference	Yes
Co-located EntityBean remote home reference, remote reference	Yes
Co-located Stateful SessionBean remote home reference	Yes
Co-located Stateful SessionBean remote reference	No
Co-located Stateless SessionBean remote home reference, remote reference	Yes
Distributed EntityBean remote home reference, remote reference	No
Distributed Stateful SessionBean remote home reference, remote reference	No

Table 18-1 Object Types Supported for J2EE Web Application Session State Failover

Java Object Type	Failover Support Provided
Distributed Stateless SessionBean remote home reference, remote reference	No
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
JMS ConnectionFactory, Destination	No
JavaMail Session	No
Connection Factory	No
Administered Object	No
Web service reference	No
Serializable Java types	Yes

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 latter 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 ONE 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 [Chapter 19, “Using the cladmin Command.”](#)

1. Create an HADB database for the cluster. This is explained in [Chapter 20, “Administering the High-Availability Database”](#) on page 499.

2. Create a session store for the session data for the HADB database for each cluster. This is explained in [“Creating a Session Store” on page 451](#).
3. 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. This is explained in [“Setting Up the JDBC Connection Pool” on page 510](#).”
4. Enable availability for the application server instances that should support session persistence. This is explained in [“Enabling Availability for an Application Server Instance” on page 456](#).
5. 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 session data is stored. This is explained in [“Setting the Persistence Type” on page 462](#).
6. If you are using the `ha` persistence type, 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 applications. This is explained in [“Setting the Persistence Frequency” on page 469](#) and [“Setting the Persistence Scope” on page 472](#).
7. If you want to fine-tune the 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 Session Persistence Properties” on page 479](#).
8. Specify the JNDI name of the JDBC resource that you created for the HADB database. This is explained in [“Specifying JNDI Name of the JDBC Resource for HADB Database” on page 481](#).
9. If you are using multiple clusters with a single session store, specify the name of the cluster to which the application server instance belongs. This is explained in [“Specifying a Cluster Name for Application Server Instance” on page 482](#).
10. Make distributable each application that should be highly available. This is explained in [“Making an Application Distributable” on page 483](#).

Creating a Session Store

This section explains how to create a session store. This section includes the following topics:

- [About Creating a Session Store](#)
- [Creating Session Stores for Multiple Clusters](#)
- [Using the `asadmin create-session-store` Command to Create a Session Store](#)

About Creating a Session Store

If you are using HADB as the persistence store, you must create a session store for each cluster to store the session data.

Before you create the session store, you must have created an HADB database. For more information, see [Chapter 20, “Administering the High-Availability Database.”](#)

Use the `asadmin create-session-store` command to create the session store for the cluster. This is an `asadmin` command that is run for the HADB database for the cluster and not for any particular application server instance. You run it only once for each cluster that you set up.

-
- | | |
|-------------|--|
| NOTE | <ul style="list-style-type: none">• You must run the <code>create-session-store</code> command before accessing any application that needs failover support in a clustered environment.• For the <code>create-session-store</code> command to work correctly, the HADB must be running. |
|-------------|--|
-

-
- | | |
|-------------|--|
| NOTE | The <code>asadmin create-session-store</code> command communicates with the HADB database and not with an application server instance. |
|-------------|--|
-

Creating Session Stores for Multiple Clusters

If you are creating multiple clusters, create a different session store for each cluster. For more information on using multiple clusters, see [“Setting up Multiple Clusters” on page 431.](#)

You can create the session stores in two ways:

- Create the session stores in the same HADB database
- Create the session store for each cluster in a separate HADB database

The table here summarizes the advantages and disadvantages of each approach. The left column lists the approach, the middle column lists the advantage(s) of the approach, and the right column lists the disadvantage(s) of the approach.

Table 18-2 Comparison of Approaches for Creating Session Stores for Multiple Clusters

Approach	Advantage(s)	Disadvantage(s)
Session stores for all clusters are created in the same HADB database.	Less resources (such as machines to host HADB active nodes and spare nodes) are required.	<ul style="list-style-type: none"> Shutting down the database for maintenance tasks affects all the clusters. More application server instances read from and write to the same HADB database. This might result in a decrease in performance.
Session store for each cluster is created in a separate HADB database.	<ul style="list-style-type: none"> Shutting down one HADB database for maintenance affects only the cluster associated with that HADB database. There is no disruption in service to other clusters. Lesser number of application server instances read from and write to the same HADB database. This can result in an increase in performance. 	More resources (such as machines to host HADB active nodes and spare nodes) are required.

NOTE 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. This is explained in [“Specifying a Cluster Name for Application Server Instance” on page 482](#).

Using the `asadmin create-session-store` Command to Create a Session Store

The syntax of the `asadmin create-session-store` command is as follows:

```
asadmin create-session-store [--storeurl persistent-store-url --storeuser
username [--storepassword userpassword] [--dbssystempassword
systempassword]] | [--optionsfile file-name]
```

This command creates a session store to be used by a particular database user.

- `storeurl` is the URL of the HADB store. To get this URL, use the following HADB command:

```
hadb_install_dir/bin/hadbm get jdbcURL database_name
```

where *hadb_install_dir* is the directory where HADB has been installed and *database_name* is the name of the HADB store. The default value of *database_name* is `hadb`.

The URL will be of the form

```
jdbc:sun:hadb:host:port,host:port,...
```

NOTE The `create-session store` command cannot run across firewalls. That is, for the `create-session-store` command to work, the application server instance and HADB must both be on the same side of a firewall.

The options of the `create-session-store` command are as follows:

- `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 20, “Administering the High-Availability Database.”](#)

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 18-3 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.

This file 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 `dbsystempassword` is `dbhapassword`.

```
asadmin create-session-store --storeurl  
jdbc:sun:hadb:myhost1:4045,myhost2:4085 --storeuser haadmin  
--storepassword hapassword --dbsystempassword 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
```

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 510](#).

Enabling Availability for an Application Server Instance

By default, availability is disabled for application server instances. This is because when you install the Sun ONE Application Server, you may not have configured the HADB, and you need the HADB for availability to work correctly. Once you have configured the HADB, you can enable availability for application server instances.

After you enable availability for an application server instance, session persistence for all the virtual servers and applications in that application server instance is enabled.

To enable availability 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 Availability Service tab.
4. Check the Availability Service Enabled box.
5. Click the Save button.

To enable availability for the instance using the command-line interface, use the `asadmin set` command to set for the instance the value of the attribute `availability-enabled` of the element `availability-service` to `true`.

Example: Enabling Availability for an Application Server Instance

The following command enables session persistence for the application server instance `server1`:

```
asadmin set --user admin_user [--password admin_password] [--host host]
[--port port] server1.availability-service.availabilityEnabled=true
```

Availability of Single Sign-on 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 185](#).

For this feature to continue to work even when a session fails over to another instance in a cluster, single sign-on information must be persisted to HADB. This is enabled when you enable availability for the application server instance.

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 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.

NOTE	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.
-------------	--

Disabling Availability for an Application Server Instance

If you disable availability for an application server instance, session persistence for all the virtual servers and applications in that application server instance is disabled. Persistence of single sign-on session state is also disabled for the application server instance.

NOTE By default, session persistence is disabled for application server instances.

To disable high availability for an application server instance using the Administration interface:

1. In the left pane of the Administration interface, under the server instance, open the Containers component.
2. Click Web Container.
3. Click the Availability Service tab under General.
4. Uncheck the Availability Service Enabled box.
5. Click the Save button.

To disable availability for the instance using the command-line interface, use the `asadmin set` command to set for the instance the value of the attribute `availability-enabled` of the element `availability-service` to `false`.

Example: Disabling availability for an Application Server Instance

In the following example, session persistence has been disabled for an application server instance called `server1`.

```
set server1.availability-service.availability-enabled=false
```

Configuring Session Persistence Options

You can configure session persistence for the Sun ONE 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 states), or you can choose to store the session after a time interval you specify (thus providing a 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 instance-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 configure-session-persistence Command](#)
- [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 Session Persistence Properties](#)

About the 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 `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. Persistence type is explained in the section [“Setting the Persistence Type” on page 462](#).
- *frequency* is the persistence frequency at which the session is stored. Persistence frequency is explained in the section [“Setting the Persistence Frequency” on page 469](#).
- *scope* is the amount of session that is stored. Persistence scope is explained in the section [“Setting the Persistence Scope” on page 472](#).
- *jdbc-resource-jndi-name* 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 456](#).
- *property* option is used to fine-tune other properties for configuring session persistence. For more information, see [“Configuring Other Session Persistence Properties” on page 479](#).
- *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 , “Using the Command Line Interface,” on [page 547](#).

Using cladmin Command to Configure Session Persistence

You can use the `cladmin` command to configure session persistence for all instances in the cluster 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 ONE Application Server (by default `etc/opt/SUNWappserver7`), you can omit the `instancefile` and `passwordfile` options.

For more information on the `cladmin` command, see [Chapter 19, “Using the cladmin Command.”](#)

Setting the Persistence Type

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`

NOTE Only the `ha` persistence type is supported for production systems *that require failover of session state*.

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 468](#).

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 session state for each application server instance in a cluster is stored in the HADB database. The session state for each instance in a cluster is available to all other instances in the cluster. Therefore, if an instance in a cluster becomes unavailable, other instances in the cluster can continue to serve the sessions that the instance that became unavailable 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 410](#).

- 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 [“Understanding Assigned Requests and Unassigned Requests” on page 426](#).
- Requests that were sent to the application server instance but that were not serviced are lost and an error is returned to the 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 17, “Managing Clusters.”](#)

When you use the `ha` persistence type, you must also configure correctly 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 469](#). For more information on setting the persistence scope, see [“Setting the Persistence Scope” on page 472](#).

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 `ha`.

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

```
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 session persistence 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 Session Persistence Properties” on page 479](#).

If the application server instance becomes unavailable unexpectedly, the corresponding session state 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`.

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">
    <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 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 ONE 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 18-8.

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 Session Persistence Properties” on page 479](#).

NOTE The limited failover capabilities offered by the `file` persistence type are intended for use only in development systems. The `file` 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`.

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">
```

```
<manager-properties>
  <property name="persistenceFrequency"
value="web-method" />
</manager-properties>
<store-properties>
  <property name="persistenceScope" value="session" />
</store-properties>
</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 18-4 Comparison of Persistence Type Options

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">In development systemsIn production systems where no failover capabilities are needed	Not applicable	Not applicable
file	No	In development systems, including to simulate failover during development without having to run HADB.	time-based	Not applicable

Table 18-4 Comparison of Persistence Type Options *(Continued)* *(Continued)* *(Continued)*

ha	Yes	In production systems where failover capabilities are required.	<ul style="list-style-type: none"> • time-based • web-method 	<ul style="list-style-type: none"> • session • modified-session • modified-attribute
----	-----	---	--	---

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.

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
```

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

```
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="memory">
  <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 Session Persistence Properties” on page 479](#).

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
```

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

```
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="memory">
  <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 18-5 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`](#)

NOTE You can choose the persistence scope only for the `ha` 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
```

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="memory">
  <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
```

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="memory">
  <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.

NOTE The persistence scope `modified-attribute` is not certified as a full production-quality feature. If you use this persistence scope, evaluate the performance and stability of the Sun ONE Application Server in expected peak load conditions. If exceptions are logged or the response time is more than that is acceptable, do not use this persistence scope for your production environment.

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
```

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

```
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="memory">
  <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. The left column lists the persistence scope option, the middle column lists the advantage(s) of the option, and the right column lists the possible disadvantage(s) of the option.

Table 18-6 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 18-6 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.	<ol style="list-style-type: none"> 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. 2. Your application must be written to meet the following constraints required by this mode: <ul style="list-style-type: none"> • Call <code>setAttribute</code> or <code>removeAttribute</code> every time you modify the session state. • Make sure there are no cross references between attributes. • Distribute the session state across multiple attributes, or at least between a read-only attribute and a modifiable attribute.

Configuring Other Session Persistence Properties

To fine tune the session persistence behavior, you can modify properties of the `manager-properties` and of `store-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 18-7](#) and [Table 18-8](#). 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 18-7 manager-properties properties Configurable for Session Persistence

Name of the Property	Default Value	Description
<code>reapIntervalSeconds</code>	60	Specifies the number of seconds between checks for expired sessions. This is also the frequency at which passivation of sessions occurs. If the persistence type is <code>file</code> or <code>ha</code> , sessions are passivated if <code>maxSessions</code> has been exceeded. If persistence frequency is time-based, active sessions are stored at this interval.
<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">• If the persistence type is <code>memory</code>, an attempt to create a new session causes an <code>IllegalStateException</code> to be thrown.• If the persistence-type is <code>file</code> or <code>ha</code>, the sessions are passivated to the persistent store.
<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 18-8 `store-properties` Configurable for Session Persistence

Property Name	Default Value	Description
directory	<i>instance_dir/generated/jsp/j2ee-apps/appname/appname_war</i>	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. Applicable only if the persistence type is <i>file</i> .

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.

```
configure-session-persistence --user admin --password MyPassword
--host localhost --port 4848 --type ha --property
maxSessions=1000:reapIntervalSeconds=60 instance1
```

Specifying JNDI Name of the JDBC Resource for HADB Database

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.

NOTE	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 Web Container.
3. Click the Availability Service tab.
4. Click Properties under Persistence Store Properties.
5. In the Name field, type the following:
`store-pool-jndi-name`
6. 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 20, “Administering the High-Availability Database.”](#)

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 504](#).

Specifying a Cluster Name for Application Server Instance

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.

To configure each cluster with a different cluster name, for each application server instance in these clusters, specify the name of the cluster to which the instance belongs.

NOTE	If you are using a separate HADB database for each cluster, or if you are using a separate session store within the same 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.

NOTE The name of the cluster need not be the same as the name of the cluster as specified in the `loadbalancer.xml` 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.

For more information on clusters, see [Chapter 17, “Managing Clusters.”](#)

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.

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
```

Making an Application Distributable

For session persistence to work for an application, the application must be *distributable*.

To make an application distributable, in the corresponding `web.xml` file, specify the `distributable` sub-element for the `web-app` element.

Example: Making an Application Distributable

The example here provides extracts from a sample `web.xml` file. The relevant portion has been **highlighted**.

```
<web-app>

    <display-name>webapps-simple</display-name>

    <description>
        A sample application.
    </description>

    <distributed></distributed>

    <servlet>

        <servlet-name>HelloWorldExample</servlet-name>
    <servlet-class>samples.webapps.simple.servlet.HelloWorldExample
        </servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>HelloWorldExample</servlet-name>
    <url-pattern>/helloworld</url-pattern>

    </servlet-mapping>

</web-app>
```

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*.

If the `web-app` element of the `web.xml` file does not have the `distributed` sub-element specified, the application is called a *non-distributable* application. A non-distributable application cannot be configured for high availability.

Default Values for Session Persistence

If high availability is disabled, persistence type is set to `memory`. 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.

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 you can use the `asadmin clear-session-store` command, disable high availability for all instances in the cluster as explained in [“Disabling Availability for an Application Server Instance” on page 459](#).

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 a Session Store” on page 451](#).

For more information on using HADB commands, see [Chapter 20, “Administering the High-Availability Database.”](#)

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 18-9 Environment Variables for the create-session-store Command Options

Option	Environment variable
--storeurl	AS_ADMIN_STOREURL
--storeuser	AS_ADMIN_STOREUSER
--storepassword	AS_ADMIN_STOREPASSWORD

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 Availability for an Application Server Instance” on page 456](#).

Using the cladmin Command

This chapter describes the use of the `cladmin` command that is part of the Sun™ Open Net Environment (Sun ONE) Application Server 7, Enterprise Edition.

This chapter contains the following sections:

- [About the cladmin Command](#)
- [asadmin Commands Supported by the cladmin Command](#)
- [Requirements and Limitations](#)
- [Input Files for the cladmin Command](#)
- [Running the cladmin Command](#)
- [Using the cladmin Command While Running Multiple Clusters](#)

About the cladmin Command

You can use the `cladmin` command to run certain `asadmin` commands simultaneously 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. This simplifies the task of cluster administration

NOTE	Only the <code>asadmin</code> commands listed in “ asadmin Commands Supported by the cladmin Command ” on page 490 have been tested and are supported although other <code>asadmin</code> commands should work.
-------------	---

It is recommended that to configure the application server instances in a cluster, you use the `cladmin` command for all the supported `asadmin` commands. This will ensure consistency in configuration of all application server instances in the cluster.

The `cladmin` command needs input files for its functioning. The information about application server instances is stored in the `clinstance.conf` file and the information about password is stored in the `clpassword.conf` file. You can modify the input files to support different configurations. Both the input files are text files. More information about the input files is provided in [“Input Files for the cladmin Command” on page 492](#).

The command reads the instance-specific information from the `clinstance.conf` file one instance at a time. You supply the rest of the argument to the `cladmin` command. The `cladmin` command invokes the corresponding `asadmin` command on each instance specified in the `clinstance.conf` file.

The command is located in `install_dir/bin` where `install_dir` is the location of the Sun One Application Server installation directory. The input files are located in the configuration directory of the Sun One Application Server. The default location of the configuration directory is `/etc/opt/SUNWappserver7`.

NOTE	The <code>cladmin</code> command 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.
-------------	--

This topic contains the following sections:

- [asadmin Commands Supported by the cladmin Command](#)
- [Requirements and Limitations](#)

asadmin Commands Supported by the cladmin Command

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 19-1 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 and Limitations

Here are important facts that you must know about the use of `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 will 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.

Input Files for the cladmin Command

The input files that are needed for the `cladmin` command to function are:

- `clinstance.conf`: This file contains information about the application server instances that are part of the cluster.
- `clpassword.conf`: This file contains the administration server password and is pre-populated with the correct password during a standard installation.

This topic contains the following sections:

- [The `clinstance.conf` File](#)
- [The `clpassword.conf` File](#)

The `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 493](#).

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*

`instancename` *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 19-2 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
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
```

The clpassword.conf File

The `clpassword.conf` file contains the administration server password. Whenever 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 administration server password.

NOTE	Permissions 0600 are preset on the <code>clpassword.conf</code> file and the file can be accessed only by the root user.
-------------	--

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 ONE 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 [“Input Files for the cladmin Command” on page 492](#).

If the input files are located in the Configuration directory of the Sun ONE 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 ONE Application Server.

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 ONE Application Server (by default `/etc/opt/SUNWappserver7`)

Example: Starting All Instances in a Cluster with Location of Input Files Specified

```
./cladmin --instancefile /tmp/clinstance.conf --passwordfile  
/tmp/clpassword.conf start-instance
```

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
```

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
```

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 19-3 Exit Codes for the cladmin Command

Exit Code	Description
0	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.

Using the cladmin Command While Running Multiple Clusters

If you are running multiple clusters, you must specify a separate `clinstance.conf` file for each cluster.

You can run the `cladmin` command on the instances in any particular cluster by specifying the path of the input files for that particular cluster as explained in [“Running the cladmin Command” on page 494](#).

Administering the High-Availability Database

This chapter describes the high-availability database (HADB) in the SunTM Open Net Environment (Sun ONE) Application Server 7, Enterprise Edition 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 20-1 HADB Roadmap

Step	Description of Task	Location of Instructions
1	Decide on your high-availability topology and set up your systems	<i>Sun ONE Application Server System Deployment Guide</i>
2	Install the HADB software with or without the Sun ONE Application Server software	<i>Sun ONE Application Server Installation Guide</i>
3	Set up the following for HADB machines: <ul style="list-style-type: none"> • Shared memory • Communication using <code>rsh</code> or <code>ssh</code> • Environment variables 	<i>Sun ONE 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*	Chapter 18, “Configuring Session Persistence”
6	Tune the HADB for maximum performance	<i>Sun ONE Application Server Performance Tuning Guide</i>

* Can be done as part of cluster setup using the `clsetup` command. See the *Sun ONE 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 ONE Application Server Installation Guide* and “Setting the Management Protocol” on page 504.

This chapter contains the following sections:

- [About the High-Availability Database](#)
- [Configuring the HADB](#)
- [Managing the HADB](#)
- [Expanding the HADB](#)
- [Monitoring the HADB](#)
- [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 `HTTPSession` information. This section includes these topics:

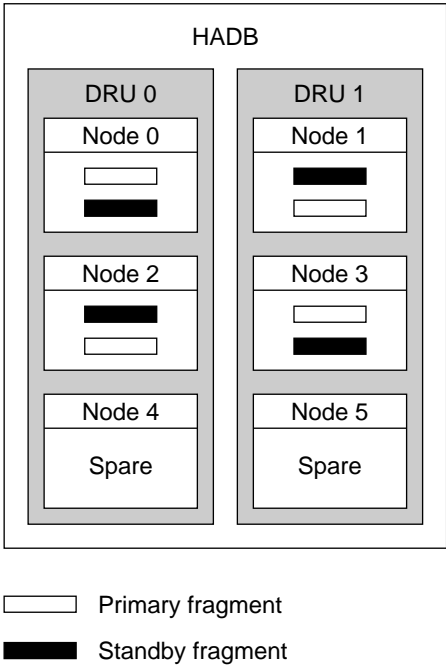
- [HADB Architecture](#)
- [HADB Nodes](#)
- [The `hadbm` Command](#)

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 20-1 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 `HTTPSession` state, including the passivated session state, is lost when a web 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 ONE 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 ONE Application Server System Deployment Guide*. For more general information about nodes and how to monitor them, see [“Node Status” on page 529](#).

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 ONE Application Server Installation Guide* and [“Setting the Management Protocol” on page 504](#).

The `hadbm` command is located in the `install_dir/SUNWhadb/4/bin` directory. 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 20-2 `hadbm` General Options

Long Form	Short Form	Description
<code>--quiet</code>	<code>-q</code>	Executes the subcommand silently without any descriptive messages.
<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.

Table 20-2 `hadbm` General Options (*Continued*)

Long Form	Short Form	Description
<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.

Configuring the HADB

This section describes the following basic HADB configuration tasks:

- [Setting the Management Protocol](#)
- [Creating a Database](#)
- [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 HADB 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 n1,n2,n3,n4,n5,n6
```

NOTE Using `ssh` is strongly recommended because `ssh` is more secure than `rsh`.

The `hadbm` command uses the `rsh` or `ssh` protocol to communicate with the database nodes. The management protocol is not used for database operations.

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 ONE Application Server Installation Guide*.

However, you can create a database outside of `clsetup`. To manually create a database to store the session data, and to start the database if `inetd` is not used, 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 n1,n2,n3,n4,n5,n6
```

If you have difficulty creating a database, check the following:

- File and directory permissions must be set to allow access to the install, history, device, and config paths for the following users:
 - Sun ONE Application Server administrative user (set during installation)
 - HADB system user

For details about setting user permissions, see the *Sun ONE Application Server Installation Guide*.

- Sun ONE 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 ONE Message Queue: 7676
 - IIOP: 3700
 - HTTP server (root): 80
 - HTTP server (non-root): 1024
 - Admin server (root): 4848
 - HADB nodes: Each node uses five consecutive ports. If the default portbase (15200) is used, node 0 uses 15200 through 15204, node 1 uses 15220 through 15224, and so on.

- Disk space must be adequate; see the *Sun ONE Application Server Installation Guide*.

Database creation errors are written to the Sun ONE Application Server’s log file and to the HADB history file. For more information about the history file, see [“Clearing and Archiving History Files” on page 540](#). If you still have difficulty creating a database, contact Sun customer support. See [“Using Sun Customer Support for the HADB” on page 542](#).

The `hadbm create` command options are listed in the following table.

Table 20-3 `hadbm create` Options

Long Form	Short Form	Default	Description
<code>--installpath</code>	<code>-n</code>	location of <code>hadbm</code> command	Specifies the HADB system installation path. This path must already exist. Use this option if the HADB server installation resides in a location different from the management client installation.
<code>--historypath</code>	<code>-t</code>	<code>/var/tmp</code>	Specifies the path to the history files. This path must already exist. For details about history files, see “Clearing and Archiving History Files” on page 540 .
<code>--devicepath</code>	<code>-d</code>	<code>/var/opt/SUNWhadb</code>	Specifies the path to the data and log devices. This path must already exist.
<code>--configpath</code>	<code>-c</code>	<code>/etc/opt/SUNWhadb</code>	Specifies the path to the configuration files used internally by the HADB. This path must already exist.
<code>--datadevices</code>	<code>-a</code>	1	Specifies the number of data devices on each node, between 1 and 8 inclusive.
<code>--portbase</code>	<code>-b</code>	15200	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 four consecutively numbered ports. If you want to run several databases on the same machine, you should have a plan for allocating port numbers and allocate them explicitly.

Table 20-3 `hadbm create` Options (Continued)

Long Form	Short Form	Default	Description
<code>--spares</code>	<code>-s</code>	0	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.
<code>--set</code>	<code>-S</code>	none	<p>Specifies a comma-separated list of database configuration attributes. For explanations of valid database configuration attributes, see “Viewing and Modifying Configuration Attributes” on page 533.</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>
<code>--inetd</code>	<code>-I</code>	not specified	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 “Additional Steps for inetd” on page 509 .
<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>--devicesize</code>	<code>-z</code>	none	<p>Specifies the size of the devices per node in MB. The device size should be as large as possible. The maximum size is the maximum operating system file size. The minimum size is derived as follows, based on the default <code>LogbufferSize</code> of 48MB:</p> $4 \times \text{LogbufferSize} + 16\text{MB} = 208\text{MB}$ <p>You can increase the device size later as described in “Adding Storage Space to Existing Nodes” on page 522.</p> <p>For more information on setting the <code>LogbufferSize</code>, see “Viewing and Modifying Configuration Attributes” on page 533.</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 “The hadbm Command” on page 502.</p>

Table 20-3 `hadbm create` Options (Continued)

Long Form	Short Form	Default	Description
<code>--dbpasswordfile</code>	<code>-P</code>	<code>none</code>	Specifies a file that stores the password to be created for the HADB system user. For details, see “The <code>hadbm</code> Command” on page 502 .
<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 item in the list. The number of nodes must be even. Using duplicate host names creates nodes on the same host with different port numbers.</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 nodes are in one DRU, even nodes in the other. If <code>--spares</code> is used, spare nodes are those with the highest numbers.</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>

NOTE

If for any reason the `hadbm create` command is not successfully executed, history files on the HADB host are deleted.

However, if the management-client machine from which the `hadbm create` command is run has a directory with the same path as the `historypath` on the HADB host, 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.

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.

NOTE If a node and its mirror fail at the same time, you must clear the database as described in [“Clearing the HADB” on page 520](#). This is called a double-node failure and `inetd` cannot help in this situation.

To set up `inetd` for a new database or a 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 your database using the `hadbm clear` command. See [“Clearing the HADB” on page 520](#).
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 515](#) and [“Getting the Status of the HADB” on page 528](#).

Setting Up the JDBC Connection Pool

The Sun ONE 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 ONE Application Server. This command is described in the *Sun ONE 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 245](#).

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 20-4 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	Other
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 20-5 HADB Connection Pool Properties

Property	Description
username	Specifies the name of the storeuser to be specified in the asadmin create-session-store command. See “Enabling Availability for an Application Server Instance” on page 456 .
password	Specifies the storepassword to be specified in the asadmin create-session-store command. See “Enabling Availability for an Application Server Instance” on page 456 .
serverList	Specifies the JDBC URL of the HADB. To determine this value, see “Getting the JDBC URL” on page 510 . You must change this value if you add nodes to the database. See “Adding Nodes to the HADB” on page 523 .
cacheDatabaseMetaData	Setting this property to false as required ensures that calls to Connection.getMetaData() make calls to the database, which ensures that the connection is valid.
eliminateRedundantEndTransaction	Setting this property to true as required improves performance by eliminating redundant commit and rollback requests and ignoring these requests if no transaction is open.

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 ONE 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\:port,host\:port,host\:port,host\:port:cacheDatabaseMetaData=false:
eliminateRedundantEndTransaction=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. For details about using escape characters, see [“Using Escape Characters” on page 558](#).

Creating a JDBC Resource

The following table summarizes JDBC resource settings required for the HADB.

Table 20-6 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 “Default Values for Session Persistence” on page 484.</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 “Creating a Connection Pool” on page 511 .
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 542](#). 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

Starting a node is not necessary if you are running with `inetd`, because `inetd` automatically restarts the node when the machine reboots.

You may want to start a node in the following circumstances:

- If you have stopped a node, for example for hardware or software replacement.
- 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.

In most cases, you should first attempt to start the node using the `normal` start level. You must use the `repair` start level in the following circumstances:

- If you are recovering from a disk crash.
- If starting a node using the `normal` start level fails or times out.

NOTE Repair of a node that does not contain data causes the repair to hang. This can happen if you attempt to start a new active node at the `repair` level before either refragmenting the database or clearing the database and recreating the session store.

To start a node in the database, use the `hadbm startnode` command. The syntax is as follows:

```
hadbm startnode [--startlevel=level] node-number [dbname]
```

For example:

```
hadbm startnode 1
```

The `hadbm startnode` command options are listed in the following table.

Table 20-7 `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.
<i>node-number</i>	<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.
<i>dbname</i>	<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 host, and you need to stop the host.

NOTE 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 528](#).

To stop a node in the database, use the `hadbm stopnode` command. The syntax is as follows:

```
hadbm stopnode [--no-repair] node-number [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.

Therefore, if you have set up `inetd`, you must perform these additional tasks when you stop a node:

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.
2. Re-add the entry to the `inetd` files once you have restarted the node.

For more information, see [“Additional Steps for inetd” on page 509](#).

The `hadbm stopnode` command options are listed in the following table.

Table 20-8 `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 takes over. By default, a spare node starts up and takes over the functioning of this node.
<i>node-number</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	<code>hadb</code>	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.

NOTE Do not restart 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 528](#).

Repair of a node that does not contain data causes the repair to hang. This can happen if you attempt to start a new active node at the `repair` level before either refragmenting the database or clearing the database and recreating the session store.

To restart a node in the database, use the `hadbm restartnode` command. The syntax is as follows:

```
hadbm restartnode [--startlevel=level] node-number [dbname]
```

For example:

```
hadbm restartnode 1
```

The `hadbm restartnode` command options are listed in the following table.

Table 20-9 `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.
<code>node-number</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>dbname</code>	<code>none</code>	<code>hadb</code>	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.

NOTE	If you stop a database from the server machine, you must also start it from the server machine. Starting it from the client machine does not work in this case.
-------------	---

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 519](#).

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 520](#).

Before stopping the HADB, you should either stop dependent Sun ONE Application Server instances or configure them to use a different persistence method. For details, see [Chapter 18, “Configuring Session Persistence”](#).

NOTE	If you stop a database from the server machine, you must also start it from the server machine. Starting it from the client machine does not work in this case.
-------------	---

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. The database you want to stop must be in one of these states: HA Fault Tolerant, Fault Tolerant, Operational, or Non-Operational. For more information about database states, see [“Getting the Status of the HADB” on page 528](#).

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 remains available. When you stop and start the HADB in separate operations, data is unavailable while the HADB is stopped. If a `hadbm set` command fails, restarting the HADB restores the previous configuration. For details about `hadbm set`, see [“Viewing and Modifying Configuration Attributes” on page 533](#).

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 or adding new nodes to a database that uses `inetd`. See [“Additional Steps for inetd” on page 509](#).
- 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 528](#).
- If you are recovering from session data corruption. See [“Recovering from Session Data Corruption” on page 542](#).

You should stop the database before clearing it. See [“Stopping the HADB” on page 518](#).

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 20-10 `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 the database uses <code>inetd</code> .
<code>--spares</code>	<code>-s</code>	0	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 “The hadbm Command” on page 502 .
<code>--dbpasswordfile</code>	<code>-P</code>	none	Specifies a file that stores the password for the HADB system user. For details, see “The hadbm Command” on page 502 .
<code>dbname</code>	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 518](#). 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 ONE 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](#)

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 530](#).

You can increase the device size per node in MB using the following `hadbm set` command:

```
hadbm set TotalDatadeviceSizePerNode=size
```

For example:

```
hadbm set TotalDatadeviceSizePerNode=2048
```

The maximum size allowed is the maximum operating system file size. The device size should be as large as possible.

Changing the `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 528](#).

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 ONE 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 ONE Application Server as described in the *Sun ONE 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 n7,n8,n9,n10
```

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 ONE Application Server System Deployment Guide*.

For details, see [“Setting Up the JDBC Connection Pool” on page 510](#).

NOTE	<p>If you created the database using <code>--inetd</code>, you must do the following:</p> <ul style="list-style-type: none">• Use the <code>--no-refragment</code> option and refragment the database in a separate step using the <code>hadbm refragment</code> command.• Take extra steps to update the <code>inetd</code> configuration files to take into account the new nodes you add. For more information, see “Additional Steps for inetd” on page 509.
-------------	---

The `hadbm addnodes` command options are listed in the following table.

Table 20-11 `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 “Refragmenting the HADB” on page 525.</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>You can add nodes without refragmenting the database; see “Adding Nodes Without Refragmenting” on page 526.</p>
<code>--spares</code>	<code>-s</code>	0	<p>Specifies the number of new spare nodes. 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>
<code>--dbpassword</code>	<code>-p</code>	none	<p>Specifies the HADB system user password. You can use <code>--dbpasswordfile</code> instead. For details, see “The hadbm Command” on page 502.</p>
<code>--dbpasswordfile</code>	<code>-P</code>	none	<p>Specifies a file that stores the password for the HADB system user. For details, see “The hadbm Command” on page 502.</p>

Table 20-11 `hadbm addnodes` Options (*Continued*)

Long Form	Short Form	Default	Description
<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	Specifies a comma-separated list of new host names for the nodes in the database. Duplicates are allowed; this creates nodes on the same host with different port numbers. One node is created for each item in the list. The number of nodes must be even.
<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 “Getting the Status of the HADB” on page 528 .

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 530](#).

If this command fails even after multiple attempts, see [“Adding Nodes Without Refragmenting” on page 526](#).

The `hadbm refragment` command options are listed in the following table.

Table 20-12 `hadbm refragment` Options

Long Form	Short Form	Default	Description
<code>--dbpassword</code>	<code>-p</code>	<code>none</code>	Specifies the HADB system user password. You can use <code>--dbpasswordfile</code> instead. For details, see “The hadbm Command” on page 502 .
<code>--dbpasswordfile</code>	<code>-P</code>	<code>none</code>	Specifies a file that stores the password for the HADB system user. For details, see “The hadbm Command” on page 502 .
<code>dbname</code>	<code>none</code>	<code>hadb</code>	Specifies the database name. The database must be in the HA Fault Tolerant or Fault Tolerant state. For more information about database states, see “Getting the Status of the HADB” on page 528 .

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. Adding nodes without refragmenting the database is not generally recommended. 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 522](#).
- 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 525](#).
- 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 [“Load Balancer Configuration File” on page 413](#).
 - b. Disable session persistence as described in [“Enabling Availability for an Application Server Instance” on page 456](#).

- 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. Clear the database as described in [“Clearing the HADB” on page 520](#).
3. Reconfigure the JDBC connection pool as described in [“Setting Up the JDBC Connection Pool” on page 510](#). You can also use the `cladmin` command. See [Chapter 19, “Using the cladmin Command.”](#)
4. Reload the session persistence store as described in [“Enabling Availability for an Application Server Instance” on page 456](#). You can also use the `cladmin` command.
5. 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 Availability for an Application Server Instance” on page 456](#).
 - 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 ONE 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 `hadbm status` command options are listed in the following table.

Table 20-13 `hadbm status` Options

Long Form	Short Form	Default	Description
<code>--nodes</code>	<code>-n</code>	not present	If present, displays node status information. See “Node Status” on page 529 .
<i>dbname</i>	<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 520](#).

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 529](#).
- State of the node. For a list of possible states and their meanings, see [“States of a Node” on page 530](#).
- 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 515](#) and [“Starting a Node” on page 514](#). If multiple nodes are in this state, clear the database as described in [“Clearing the HADB” on page 520](#).
- **Running:** The node is fully up and running, providing services 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 restarting from main memory or from disk. It uses the log files from the mirror node and recovers the lost data. 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 files 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 ONE 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 ONE Application Server System Deployment Guide* and the *Sun ONE 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 buffer 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 available locks should be used. To change the `NumberOfLocks`, see [“Viewing and Modifying Configuration Attributes” on page 533](#).

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 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

Viewing and Modifying Configuration Attributes

You can modify database configuration variables. 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 534](#)), 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 534](#)), 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 528](#).

If execution of this command is unsuccessful, restart the HADB as described in [“Restarting the HADB” on page 519](#).

NOTE	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 doesn't take effect until a new HADB connection is made from a Sun ONE 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 20-14 Configuration Attributes

Attribute	Default	Range	Description
ConfigPath	<code>/etc/opt/SUNWhadb</code> <code>/dbdef</code>		(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.

Table 20-14 Configuration Attributes (*Continued*)

Attribute	Default	Range	Description
DevicePath	/var/tmp		(get only) Location of the data and log devices.
EagerSessionThreshold	50 (% of NumberOfSessions)		<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		Amount of time a database connection can be idle before it expires when eager session expiration is used.

Table 20-14 Configuration Attributes (*Continued*)

Attribute	Default	Range	Description
EventBufferSize	0	0MB - 2GB	Size of the event buffer, where database events are logged. During failures, the event buffer is dumped. This gives valuable information on the cause of the failures and is useful during trial deployment. Writing events to memory has a performance penalty.
HistoryPath	/var/tmp		(get only) Location of the HADB history files, which contain information, warnings, and error messages.
InstallPath	location of hadbm command		(get only) Location of the HADB executable files.
InternalLogbufferSize	12	4MB - 128MB	Size of the node internal log buffer, 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 hadbm command and the database nodes.
MaxTables	1100	100 - 1100	Maximum number of tables allowed in an HADB database.

Table 20-14 Configuration Attributes (*Continued*)

Attribute	Default	Range	Description
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		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 SHORT , login and logout of SQL sessions are recorded. If FULL , all SQL queries being prepared and executed, including parameter values, are recorded.

Table 20-14 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 - 600	<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>

Table 20-14 Configuration Attributes (*Continued*)

Attribute	Default	Range	Description
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 syslog 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 host.</p> <p>Set to none to disable syslog logging.</p>
SysLogging	TRUE		<p>Set to true if an HADB node should write any information to the operating system's syslog files.</p>
SysLogLevel	warning	none, alert, error, warning, info	<p>Lets you to fine tune which HADB messages are reported to the operating system's syslog files.</p> <p>Higher levels include lower levels. For example, the error level includes alert messages, and the info level includes all messages.</p>
SyslogPrefix	HADB		<p>Text string that is prepended to all syslog messages written by the HADB.</p>

Table 20-14 Configuration Attributes (*Continued*)

Attribute	Default	Range	Description
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 maximum operating system file size, minimum is: 4 x LogbufferSize + 16MB HADB uses the minimum space as defined here for internal use and, therefore, this space is not available for use on the data device.	Sum of the sizes of all data devices used by an HADB node.

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.nodenum`. For details about `hadbm create`, see [“Creating a Database” on page 505](#).

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 542](#).

Recovering from Session Data Corruption

The following are indications that session data may be corrupted:

- Error messages appear in the Sun ONE 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 485](#).
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 520](#).
3. If clearing the database doesn’t work, delete and then recreate the database. See [“Removing a Database” on page 521](#) and [“Creating a Database” on page 505](#).

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
- Host 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 network interfaces per host
- HADB data
 - History files
 - Version information

Environment Variables

This table lists environment variables that correspond to `hadbm` command options.

Table 20-15 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>
<code>--dbpassword</code>	<code>-p</code>	<code>none</code>	<code>\$HADB_DBPASSWORD</code>

Table 20-15 HADB Options and Environment Variables (*Continued*)

Long Form	Short Form	Default	Environment Variable
--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	location of hadbm command	\$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”

Appendix C, “Compiling and Configuring Apache Web Server”

Appendix D, “Running Sun ONE Message Queue Broker with
Trial Enterprise License”

Appendix E, “Third Party Copyright Notices.”

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 ONE Application Server.

For more information on the Ant tasks, see the *Sun ONE 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 ONE 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 ONE Application Server Developer's Guide to Clients</i> .
verifier	Validates the J2EE deployment descriptors with the DTDs. For more information, see the <i>Sun ONE Application Server Developer's Guide</i> .

Table A-1 Other Command-Line Utilities

Utility	Definition
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.
clsetup	Automates the process of setting up a cluster in a typical single machine configuration.
cladmin	Runs certain asadmin commands simultaneously on all application server instances in a cluster.

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. 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 ONE Application Server Administrator's Configuration File Reference*.

For more information on individual commands, see [“Command Reference” on page 565](#) 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 the “[Short and Long Options, Default Values, and Environment Variables](#)” table in “[Command Reference](#)” on page 565.

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 553. 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 597.

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 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 the [“Environment Variables Used with export Command”](#) table.

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 597):

- --host
- --port
- --user
- --password or --passwordfile
- --secure=true (if secure)
- --instance (if needed)

The following table, “[Environment Variables Used with export Command](#)”, 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 597

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 localhost.
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	=true if secure.
AS_ADMIN_INSTANCE	Sets the instance of the Sun ONE 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 ONE 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 551](#).

For more information on using commands interactively, see [“Using Interactive and Non-Interactive Options” on page 553](#).

Example from Command Line

```
> asadmin create-instance --user admin --password password --host
austen --port 4848 --instanceport 1024 server2
```

After the command completes, you return to the operating system prompt.

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
```

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`.

NOTE You do not need to use escape characters for colons in the `get` and `set` 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 Solaris, you can use either two backslashes (\\) or double-quotes (“ ”) to escape restricted characters.

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_lo
okup_app: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_a
pp:password=staging_lookup_app OraclePoollookup
```

You can also use the methods described in [“Escape Characters on All Platforms in Singlemode”](#) on page 559.

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_look
up_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
:password=staging_lookup_app OraclePoollookup
```

Using get and set Commands

Use the `get` and `set` commands to access and change configuration settings in the Sun ONE 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 <code>reconfig</code> in order for the changes to be applied to the server. For more information on applying changes/reconfiguring the server, see “Applying Changes to an Application Server Instance” on page 81.

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.mimel.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.monitoringEnabled=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 ONE Application Server, see [“Extracting Monitoring Data Using the CLI” on page 130](#).

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.

You can also access the command-line help pages as manpages in a UNIX environment. For an unbundled installation, add `install_dir/man` to your `MANPATH` environment variable. Once this is done, you can access manpages for Sun ONE Application Server utilities, for example, by typing `man asadmin` at the command prompt.

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
```

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 553](#).

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 555](#).

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 ONE Application Server. For more information regarding security in the Sun ONE Application Server, see the *Sun ONE 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 71 .

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 left column shows the command name, and the right column shows its use.

Table A-4 `asadmin` Commands

Command	Use
add-resources	Adds one or more resources of type jdbc, jms, or javamail.

Table A-4 asadmin Commands

Command	Use
clear-session-store	Clears the sessions for all application server instances from the HADB database.
configure-session-persistence	Configures session persistence options for an application server instance.
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.

Table A-4 asadmin Commands

Command	Use
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.
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-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.

Table A-4 `asadmin` Commands

Command	Use
<code>deploydir</code>	Deploys an EJB, WEB, connector, appclient, or application component that is in the directory to the application server instance.
<code>disable</code>	Disables a deployed component in the application server instance.
<code>display-license</code>	Displays license information. This command can only be executed locally.
<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-listeners</code>	Lists IIOP listeners for a server instance.
<code>list-javamail-resources</code>	Lists Java Mail resources for a server instance.

Table A-4 asadmin Commands

Command	Use
list-jdbc-connection-pools	Lists JDBC connection pools for a server instance.
list-jdbc-resources	Lists JDBC resources for a server instance.
list-jmsdest	Lists JMS destinations for a server instance.
list-jms-resources	Lists JMS resources for a server instance.
list-jndi-resources	Lists JNDI resources for a server instance.
list-lifecycle-modules	Lists lifecycle modules for a server instance.
list-mimes	Lists MIME types files for a server instance.
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.

Table A-4 `asadmin` Commands

Command	Use
<code>stop-instance</code>	Stops the server instance.
<code>undeploy</code>	Removes the deployed component from the server instance.
<code>unset</code>	Unsets the exported environment variables for <code>asadmin</code> .
<code>update-file-user</code>	Updates an existing file realm user.
<code>version</code>	Displays version information for the Sun ONE 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	jms-service
Transaction service configuration	transaction-service
MDB container configuration	mdb-container
EJB container configuration	ejb-container
Web container configuration	web-container
JVM configuration	java-config
ORB configuration	orb or iiop-service
ORB listener configuration	orblistener or iiop-listener
	Note that orblistener or iiop-listener are not valid names as is; both require the names of the listener to follow. For example:
	ORB listener configuration orblistener.<listener name> or iiop-listener.<listener name>
Log configuration	log-service
Security configuration	security-service
HTTP configuration	http-service

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	jdbc-resource
JNDI resource configuration	jndi-resource
JDBC connection pool resource configuration	jdbc-connection-pool
Custom resource configuration	custom-resource
JMS resource configuration	jms-resource

Table A-6 Resource Names for Command-Line Interface

Resource	Dotted Name
Persistence manager factory resource configuration	persistence-manager-factory-resource
Java mail resource configuration	mail-resource

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	application
EJB module configuration	ejb-module
Web module configuration	web-module
Connector module configuration	connector-module

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	http-listener or http-server.http-listener
MIME types file	mime
ACL	acl
Virtual server	virtual-server
Authentication databases	auth-db
Security realms	authrealm

Table A-8 Other Item Names for Command-Line Interface

Item	Dotted Name
Lifecycle module	lifecycle-module
Profiler configuration	profiler
Server configuration	server configuration (name of server instance)

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

Table A-12 EJB Container Attributes

server.xml Name	asadmin Name
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
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

server.xml Name	asadmin Name
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
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
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.orblister.orblister_id.*
```

or

```
asadmin> get server1.iiop-listener.orblister_id.*
```

To get an attribute called port:

```
asadmin> get server1.orblister.orblister_id.port
```

or

```
asadmin> get server1.iiop-listener.orblister_id.port
```

To set an attribute called address to bluestar:

```
asadmin> set server1.orblister.orblister_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
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

Table A-18 Security Realm Configuration Attributes

server.xml Name	asadmin Name
default-principal	defaultPrinicpal
default-principal-password	defaultPrinicpalPassword
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

Table A-21 JNDI Resource Attributes

server.xml Name	asadmin Name
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

Table A-22 JDBC Connection Pool Attributes

server.xml Name	asadmin Name
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
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

server.xml Name	asadmin Name
name	name
location	location
description	description

Table A-28 EJB Module Attributes

server.xml Name	asadmin Name
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/repos
itory/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="/ex
port/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

Table A-31 HTTP Listener Attributes

server.xml Name	asadmin Name
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
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_1.*
```

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.sharedpassword.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
```

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

Table A-39 Server Configuration Attributes

server.xml Name	asadmin 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_THREAD S
acls	--acls			AS_ADMIN_ACLS
address	--address			AS_ADMIN_ADDRESS
adminpassword	--adminpassword			AS_ADMIN_ADMINPASSWD

Table A-40 Short and Long Options, Default Values, and Environment Variables

Option Name	Long Format	Short Format	Default Value	Environment Variable
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
discardmanualchanges	--discardmanualchanges	-d	false	AS_ADMIN_DISCARDMANUALCHANGES
echo	--echo	-e	false	AS_ADMIN_ECHO
enabled	--enabled			AS_ADMIN_ENABLED
enforcebwlimit	--enforcebwlimit			AS_ADMIN_ENFORCEBWLIMIT
enforceconnlimit	--enforceconnlimit			AS_ADMIN_ENFORCECONNLIMIT
failconnection	--failconnection		false	AS_ADMIN_FAILCONNECTION

Table A-40 Short and Long Options, Default Values, and Environment Variables

Option Name	Long Format	Short Format	Default Value	Environment Variable
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
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

Table A-40 Short and Long Options, Default Values, and Environment Variables

Option Name	Long Format	Short Format	Default Value	Environment Variable
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_TLSEENABLED
tlsrollbackenabled	--tlsrollbackenabled			AS_ADMIN_TLSROLLBACKENABLED
transprotocol	--transprotocol		smtp	AS_ADMIN_TRANSPROTOCOL
type	--type			AS_ADMIN_TRANSPROTOCOLCLASS
upload	--upload	-U	true	AS_ADMIN_TYPE
url	--url			AS_ADMIN_URL
user	--user	-u		AS_ADMIN_USER

Table A-40 Short and Long Options, Default Values, and Environment Variables

Option Name	Long Format	Short Format	Default Value	Environment Variable
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

The scenarios in this appendix illustrate how session persistence settings for application server instances and web applications affect the failover of session state in Sun™ Open Net Environment (Sun ONE) Application Server 7, Enterprise Edition.

This appendix includes 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 session persistence is relevant to the scenarios described in this appendix:

- For production environments that require HTTP session persistence and failover capabilities, the only persistence type supported is `ha`.

- For an application server instance to support session persistence, it has to be enabled for high availability. For more information about enabling high-availability for an application server instance, see [“Enabling Availability for an Application Server Instance” on page 456](#). In the scenarios described here, it is assumed that high-availability has been enabled for the application server instances.
- For a web application to take part in session persistence, it should be distributable. For more information about distributable web applications, see [“Making an Application Distributable” on page 483](#). 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 18, “Configuring Session Persistence.”](#)

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 sessions have been created becomes unavailable, the 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 shutdown. When the instance restarts, it can recover 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 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 session state to the high-availability database (HADB).

If the persistence frequency for this application server instance is set to `time-based`, the 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 Session Persistence Properties” on page 479](#).

If the persistence frequency for the application server instance is set to `web-method`, the session state is stored after every web request.

Thus, the persistence frequency setting has a bearing on how up to date is the session state in the HADB at any time.

The application server instances to which the sessions are failed over access the session state from the HADB.

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 604](#).

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 606](#).

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 607.

Compiling and Configuring Apache Web Server

This section describes how you should compile the Apache source code and configure your installation of Apache Web Server to use the load balancer plug-in.

- [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-0.9.6g` (source)
- `mod_ssl-2.8.14-1.3.27` (source)
- `apache_1.3.27` (sources)
- `gcc-3.3-sol9-sparc-local` packages (for Solaris 9).
- `flex-2.5.4a-sol9-sparc-local` packages (for Solaris 9)
- Sun ONE Application Server 7 Enterprise Edition installables.
- 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.
 - To install `gcc 3.3`, run `pkgadd -d gcc-3.3-sol9-sparc-local`.

- To install flex, run `pkgadd -d flex-2.5.4a-sol9-sparc-local`.

NOTE These software sources are available at <http://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-0.9.6g` source and follow these steps.

1. `cd openssl-0.9.6g`
2. `./config`
3. `make`

Configuring Apache with mod SSL

Unpack the `mod_ssl-2.8.14-1.3.27` source and follow these steps.

1. `cd mod_ssl-2.8.14-1.3.27`

2.

```
run ./configure
--with-apache=../apache_1.3.27--with-ssl=../openssl-0.9.6g
--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_1.3.27` source distribution.

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. Decompress 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. Compile Apache using the `make` command, as follows:

- a.

```
cd apache_1.3.27
```
- b.

```
make
```
- c.

```
make certificate
```
- d.

```
make install
```

NOTE

- This installs apache in the location provided in the `--prefix` attribute in [“Configuring Apache with mod SSL” on page 612](#).
- `make certificate` would ask for a secure password. Remember this password as it's required for starting secure Apache.

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.

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 Sun ONE Message Queue Broker with Trial Enterprise License

Sun™ Open Net Environment (Sun ONE) Application Server 7, Enterprise Edition uses the Sun™ Open Net Environment (Sun ONE) Message Queue as the JMS provider.

This appendix describes how to run the Sun ONE Message Queue (MQ) broker using a trial enterprise license so that you can evaluate the enterprise features of the MQ broker Enterprise Edition.

About MQ

The Platform Edition of MQ is bundled with the Sun One Application Server.

The Platform Edition of MQ does not include the following features that are offered by the Enterprise Edition of MQ.

- 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.

Running MQ Broker with Trial License of Enterprise Edition of MQ

To run the MQ broker with a trial license of the Enterprise Edition of MQ, start the MQ broker that is used by the application server instance with the following additional option:

```
-license try
```

NOTE By default, when an application server instance is started, a MQ broker instance is also started.

Example: Running the MQ Broker with Trial License of Enterprise Edition of MQ

The following example shows how you can use the `asadmin` command to run the MQ broker with the trial license of the Enterprise Edition of MQ. 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
```

NOTE Enclose the string `server1.jms-service.startArgs= -license try` in double-quotes, as shown in the example.

NOTE 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.”](#)

For the change to take effect, manually apply the configuration and restart the application server instance.

Purchasing the Enterprise Edition License of MQ

After the evaluation period, if you want to purchase the Enterprise Edition license, you can purchase it at:

<http://store.sun.com/catalog/doc/BrowsePage.jhtml?catid=62774>

Once you have installed the purchased Enterprise Edition license, you no longer need to use the `-license try` option to run the MQ broker with the Enterprise Edition of MQ.

Third Party Copyright Notices

This product includes code licensed from RSA Security, Inc.

Portions of this product were developed using ANTLR. ANTLR 1989-2000 developed by jGuru.com, <http://www.ANTLR.org> and <http://www.jGuru.com>.

This product includes software developed through the Netbeans Project at <http://www.netbeans.org> under the Sun Public License. Such software, if available, may be found at www.netbeans.org.

This product includes Perl. A copy of Perl, if available, may be found at <http://public.ActiveState.com/gsar/APC/>.

This product includes software developed through the Exolab Project (<http://www.exolab.org>).

This product includes software developed through the DOM4J Project (<http://dom4j.org/>).

This product includes software developed by Apache Foundation. Copyright (c) 1999-2001 The Apache Software Foundation. All rights reserved.

This product includes software developed by The Regents of University of California. Copyright (c) 1991, 1993 The Regents of University of California. All rights reserved.

This product includes software developed by International Business Machines Corporation. Copyright (c) 1995-2001 International Business Machines Corporation and others. All rights reserved. The IBM code was obtained under the ICU License. See below.

ICU License - ICU 1.8.1 and later.

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1995-2001 International Business Machines Corporation and others
All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

All trademarks and registered trademarks mentioned herein are the property of their respective owners.

Glossary

This glossary provides definitions for common terms used to describe the Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition. *See also* [general ACL](#).

activation The process of transferring an enterprise bean's state from secondary storage to memory.

Administration interface The set of browser based forms used to configure and administer the Sun ONE Application Server 7, Enterprise Edition. *See also* [CLI](#).

administration server An application server instance dedicated to providing the administrative functions of the Sun ONE Application Server 7, Enterprise Edition, 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 ONE 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 ONE 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.

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 ONE 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 ONE 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 ONE 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 ONE Application Server 7, Enterprise Edition. *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 ONE Application Server 7, Enterprise Edition 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 source A handle to a source of data, such as a database. Data sources are registered with the iPlanet 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 ONE 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 ONE Application Server 7, Enterprise Edition, 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.

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.

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 ONE 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 \(HADB\)](#).

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 ONE Application Server uses the HADB to store HTTP session state.

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 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 ONE Directory Server, a provided LDAP server, you can store all of your enterprise's information in a single, centralized repository of directory information that any application server can access through the network.

LDIF LDAP Data Interchange Format. Format used to represent Sun ONE 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.

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 ONE Application Server 7, Enterprise Edition 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 session state is stored in the Sun Open Net Environment (Sun ONE) 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 ONE Application Server 7, Enterprise Edition, 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 ONE Application Server 7, Enterprise Edition 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 ONE 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 ONE Application Server 7, Enterprise Edition, 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 ONE Application Server 7, Enterprise Edition 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.

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 ONE Application Server 7, Enterprise Edition 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 ONE Directory Server The Sun ONE version of Lightweight Directory Access Protocol (LDAP). Every instance of Sun ONE Application Server 7, Enterprise Edition uses Sun ONE Directory Server to store shared server information, including information about users and groups. *See also* [LDAP](#).

Sun ONE Message Queue The Sun ONE enterprise messaging system that implements the Java Message Service (JMS) open standard: it is a JMS provider.

system administrator The person who administers Sun ONE Application Server 7, Enterprise Edition software and deploys Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server also supports some non-Java web application technologies, such as SHTML and CGI.

web cache An Sun ONE Application Server 7, Enterprise Edition 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 ONE Application Server 7, Enterprise Edition.

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 web browsers.

Web Server Plugin The web server plugin is an HTTP reverse proxy plugin that allows you to instruct a Sun One Web Server or Sun ONE Application Server 7, Enterprise Edition 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.

A

- accept language header, parsing 364
- acceptor threads
 - specifying number via HTTP listener 57
 - virtual servers 353
- access 120
- access control, using virtual servers 358
- access log files 104, 117
 - configuring 120
 - rotation 104
 - viewing 117
- access.log 90
- ACL, attributes 592
- acl, dotted name 592
- activation, definition 189
- Adaptor, Resource 203
- additional document directories 378
- add-resources command 308, 565
- administered objects *See* JMS administered objects
- Administration interface
 - accessing 39, 50
 - administering transactions 215
 - automatic transaction recovery 206
 - configuring log service attributes 113
 - general settings, configuring 83
 - JVM options, configuring 85
 - JVM Profiler, configuring 85
 - online help, accessing 43
 - path settings, configuring 84
 - shutting down the Administration Server 52
 - standard buttons 42
 - tabs, using 42
 - using 39
- Administration Server
 - about 50
 - applying changes 56
 - control settings, viewing 56
 - settings, accessing 54
 - shutting down, methods for 52
 - starting the SNMP master agent 176
 - starting, methods for 51
- administration, tools and associated functions 36
- administrative domains
 - about 59
 - creating 37
- admin-service 101
- afterBegin 213
- afterCompletion 213
- agents, SNMP 167
- ALERT 98
- analyzer, log
 - running (archive server logs prior to use) 121
- ansi_x3.4-1968 387
- ansi_x3.4-1986 387
- Ant tasks 548
- applient utility 548
- application
 - deploying on cluster 432
 - disabling in cluster 432, 437
 - enabling in cluster 432
 - quiescing in cluster 437
 - redeploying on cluster 441
 - specifying quiescing period for 433

- undeploying from cluster [440](#)
- application and server log output, redirecting [103](#)
- Application Client Container (ACC)
 - client side logging [102](#)
- application client JAR file [322](#)
- Application Server
 - logging features and functions [89](#)
 - overview and key features [35](#)
- application server instance
 - about [70](#)
 - accessing [46](#)
 - advanced settings [87](#)
 - applying changes [81](#)
 - starting and stopping [71](#)
 - starting manually [76](#)
 - status, viewing [83](#)
- application, dotted name [587](#)
- application.xml deployment descriptor [324](#)
- application-client.xml deployment descriptor [324](#)
- applications
 - attributes [587](#)
 - connection sharing [265](#)
 - directory structure [326](#)
 - disabling [332](#)
 - dynamic reloading [332](#)
 - elements of web [183](#)
 - environment entries [227](#)
 - J2EE, introduction [323](#)
 - JMS and [284](#)
 - JNDI lookup names [326](#)
 - monitoring object type [136](#)
 - naming standards [325](#)
 - resource environment references [238](#)
 - resource references [237](#)
 - runtime environment [328](#)
- appserv.mib [160](#)
 - managed objects and descriptions [161](#)
- appservd [79](#)
- appserv-wdog [78](#)
- archiving, log files [103](#)
- AS_ADMIN_HOST [555](#)
- AS_ADMIN_INSTANCE [555](#)
- AS_ADMIN_PASSWORD [555](#)
- AS_ADMIN_PORT [555](#)
- AS_ADMIN_PREFIX [561](#)
- AS_ADMIN_SECURE [555](#)
- AS_ADMIN_USER [555](#)
- asadmin utility
 - about [548](#)
 - attributes [570](#)
 - command syntax [550](#)
 - commands [550](#)
 - concurrent access [565](#)
 - database, administering and monitoring
 - transactions [218](#)
 - default values [597](#)
 - dotted names [570](#)
 - environment commands [553](#)
 - environment variables [597](#)
 - escape characters [558](#)
 - exit status [563](#)
 - export [553](#)
 - extracting monitoring data [130](#)
 - from command line [557](#)
 - from pipe [558](#)
 - from script [558](#)
 - get [560](#)
 - help [563](#)
 - interactive [553](#)
 - JVM settings [86](#)
 - license commands [47](#)
 - local [556](#)
 - long options [597](#)
 - multimode [552](#)
 - non-interactive [553](#)
 - operands [551](#)
 - options [550](#)
 - password file option [555](#)
 - reconfig [560](#)
 - remote [556](#)
 - restarting instances [77](#)
 - security [564](#)
 - set [560](#)
 - short options [597](#)
 - singlemode [552](#)
 - starting and stopping instances [72](#)
 - transaction administration [218](#)
 - unset [554](#)
 - usage [564](#)
- ascii [387](#)
- atomicity [202](#)

- attributes
 - EJB container (that can be monitored) 196
 - transaction, deployment descriptors 211
 - transactions 208
 - virtual server 184
 - web-module 184
- attributes, about 415
- attributes, asadmin 570
- auth-db 594
- authentication 313
- authentication database attributes 594
- authorization realm attributes 594
- authrealm 594
- AuthTrans qos-handler 156
- auto-commit connection validation 264
- availability
 - disabling for application server instances 459
 - enabling for application server instances 456
- avax.transaction.UserTransaction 214

B

- bean-cache
 - monitoring attribute names 140
 - monitoring object type 137
- beanIdleTimeoutInSeconds 196
- bean-managed transactions
 - not allowed for entity beans 214
- bean-method
 - monitoring attribute names 141
 - monitoring object type 137
- bean-pool, monitoring object type 137
- beans, message-driven
 - characteristics 193
- beforeCompletion 213
- boolean options 551
- business methods, transactions 211, 213

C

- cache control directives, setting 390
- cache settings, configuring EJB 197
- CacheBucket
 - monitoring attributes 148
 - monitoring HTTP server elements 143
- cacheDatabaseMetaData property 512
- cacheFaultsPercentage 196
- cache-hits 141
- cache-misses 140
- cache-resize-quantity 140
- capture-schema utility 548
- CGIs 385
 - settings for virtual servers 365
 - with virtual servers 359
- character set
 - changing 386
 - iso_8859-1 387
 - us-ascii 387
- chroot settings 365
- ciphers, TLS Rollback option 319
- cladmin command
 - exit codes 496
 - input files 490, 492
 - location 490
 - location of input files 490
 - log files 496
 - messages 496
 - requirements and limitations 491
 - running 494
 - supported asadmin commands 490
 - syntax 494
 - using while running multiple clusters 497
- classpathprefix 86
- client name mapping example 133
- clients
 - lists of accesses 120
- clinstance.conf file 490, 492
 - entries in 493
 - format 492
 - location 490
 - sample file 493
- clpassword.conf file 490, 494
 - format 494

- location [490](#)
- permissions [494](#)
- cluster
 - about [424](#)
 - about adding application server instances to [427](#)
 - adding application server instances to [430](#)
 - defining [426](#)
 - deploying web application on [432](#)
 - disabling web application in [432](#)
 - enabling web application in [432](#)
 - reconfiguring application server instance in running cluster [440](#)
 - redeploying web applications on [441](#)
 - removing [443](#)
 - removing application server instance from [442](#)
 - requirements for adding application server instances to [429](#)
 - sample configuration [433](#)
 - setting up multiple [431](#)
 - specifying name for session persistence configuration [482](#)
 - starting [434](#)
 - stopping [442](#)
 - stopping application server instance in [441](#)
 - time interval for health check [427](#)
 - timeout value for health check [427](#)
 - undeploying web applications from [440](#)
 - URL for health checker [427](#)
 - using multiple clusters for online upgrade [439](#)
- cluster element [417](#)
- command line, asadmin from [557](#)
- command-line interface
 - name mapping, monitoring [132](#)
 - shutting down the Administration Server [53](#)
 - starting the Administration Server [51](#)
- commands
 - asadmin [550](#)
 - license [47](#)
- commits *See* transactions, commits
- community string, with SNMP agent [166](#)
- components, MDB, *See* MDB
- concurrent access, asadmin [565](#)
- concurrent connections
 - virtual servers, quality of service [158](#)
- CONFIG [98](#), [168](#), [172](#), [173](#)
 - master agent, editing [173](#)
- configuration files, about [46](#)
- configure-session-persistence [460](#)
- connection factories
 - defined [232](#)
 - JNDI [232](#)
 - URL [238](#)
- Connection object [232](#)
- connection pool
 - properties [512](#)
- connection pooling
 - about [263](#)
 - datasource object [249](#)
- connection sharing [265](#)
- connection validation [264](#)
- Connection Validation Required setting [511](#)
- ConnectionQueue [149](#)
 - monitoring ConnectionQueueBucket attributes [146](#)
 - monitoring HTTP server elements [143](#)
- ConnectionQueue attributes
 - monitoring [144](#)
- ConnectionQueueBucket
 - monitoring HTTP server elements [143](#)
- ConnectionQueueBucket monitoring attributes [146](#)
- connections, shareable or non-shareable [207](#)
- connector modules
 - attributes [589](#)
 - deployment directory structure [327](#)
- connector-module [589](#)
- container-managed transactions [207](#)
- containers
 - EJB, responsibilities [190](#)
 - MDB [286](#)
 - Web, about [181](#)
- context-root [184](#)
- context-root attribute [419](#)
- control settings, viewing Administration Server [56](#)
- conversational state [191](#)
- CORBA, about [311](#)
- COSNaming service [230](#)
- Count200 through Count503 [150](#)
- Count2xx through Count5xx [150](#)
- CountAsyncAddrLookups [147](#)

- CountAsyncLookupsInProgress [147](#)
- CountAsyncNameLookups [147](#)
- CountBytesReceived [150](#)
- CountBytesTransmitted [150](#)
- CountCacheEntries [147](#)
- CountCacheHits [147](#)
- CountCacheMisses [147](#)
- Countcalls [151](#)
- CountConfigurations
 - monitoring Process attributes [145](#)
- CountConnections [147](#)
- CountContentHits [149](#)
- CountContentMisses [149](#)
- CountEntries [148](#)
- CountFlushes [148](#)
- CountHits [148](#)
- CountInfoHits [149](#)
- CountInfoMisses [149](#)
- CountMisses [149](#)
- CountOpenConnections [150](#)
- CountOpenEntries [148](#)
- CountOther [150](#)
- CountOverflow
 - monitoring ConnectionQueueBucket
 - attributes [146](#)
- CountQueued [146](#)
 - monitoring ConnectionQueueBucket
 - attributes [146](#)
- CountRefusals [148](#)
- CountRequests [150, 151](#)
- CountThreads [146](#)
- CountThreadsIdle [146](#)
- CountTimeouts [148](#)
- CountTotalConnection
 - monitoring ConnectionQueueBucket
 - attributes [146](#)
- CountTotalQueued
 - monitoring ConnectionQueueBucket
 - attributes [146](#)
- cp367 [387](#)
- cp819 [387](#)
- create-acl command [566](#)
- create-authdb command [566](#)

- create-auth-realm command [566](#)
- create-custom-resource command [566](#)
- create-domain command [61, 566](#)
- create-file-user command [566](#)
- create-http-listener command [360, 566](#)
- create-http-qos command [347, 366, 566](#)
- create-iiop-listener command [566](#)
- create-instance command [79, 566](#)
- create-javamail-resource command [566](#)
- create-jdbc-connection-pool command [260, 566](#)
- create-jdbc-resource command [250, 262, 566](#)
- create-jmsdest command [308, 566](#)
- create-jms-resource command [309, 566](#)
- create-jndi-resource command [566](#)
- create-jvm-options command [86, 566](#)
- create-lifecycle-module command [337, 566](#)
- create-mime command [349, 566](#)
- create-persistence-resource command [566](#)
- create-profiler command [566](#)
- create-session-store
 - environment variables [455](#)
- create-ssl command [566](#)
- create-virtual-server command [362, 567](#)
- cron [104](#)
 - scheduling execution of logadm [110](#)
- crontab, entry format [109](#)
- custom resources
 - about [233](#)
 - attributes [584](#)
 - creating [233](#)
- custom-resource [584](#)

D

- daemon
 - native SNMP, restarting [169](#)
- Data Source Enabled setting [513](#)
- data store [240](#)
- databases
 - administering and monitoring via CLI [218](#)
 - connection validation [257](#)

- JDBC API 246
- JNDI names 224
- naming services 254
- resource manager 203
- resource references 225
- three-tier access model 247
- two-tier access model 247
- DataSource 232
- DataSource object 248
- dbssystempassword 454
- debug 74
- debug mode
 - starting the application server instance 73
- default handlers
 - subsystem logging 101
- default HTTP listener
 - Administration Server 57
 - HTTP server 359
- default option values 597
- default web module 365
- delete-acl command 567
- delete-authdb command 567
- delete-auth-realm command 567
- delete-custom-resource command 567
- delete-domain command 63, 567
- delete-file-user command 567
- delete-http-listener command 362, 567
- delete-http-qos command 347, 366, 567
- delete-iiop-listener command 567
- delete-instance command 80, 567
- delete-javamail-resource command 567
- delete-jdbc-connection-pool command 567
- delete-jdbc-resource command 567
- delete-jmsdest command 309, 567
- delete-jms-resource command 309, 567
- delete-jndi-resource command 567
- delete-jvm-options command 86
- delete-lifecycle-module command 337, 567
- delete-mime command 349, 567
- delete-persistence-resource command 567
- delete-profiler command 567
- delete-ssl command 567
- delete-virtual server command 567
- delete-virtual-server command 368
- deploy command 333, 567
- deploydir command 334, 568
- deploying
 - COSNaming service with 231
 - directory structure 326
 - disabling 332
 - dynamic 331
 - EJB JAR modules 336
 - individual modules 336
 - lifecycle modules 337
 - redeployment 331
 - RMI/IIOP clients 338
 - runtime environments 328
 - using Administration interface 335
 - using asadmin 333
 - using Sun ONE Studio 335
 - WAR modules 336
 - web application on cluster 432
- deployment descriptors
 - entries 275
 - J2EE standard 323
 - Sun ONE Application Server 324
 - transaction attributes 211
- deployment, hot
 - deploying an application at server runtime, without restart 186
- description element 417
- destinations, for JMS messages, *See* JMS destinations
- directives, configuring logging 116
- directories, additional document 378
- directory structure, deployment 326
- disable command 568
- disable-timeout-in-minutes attribute 419
- disabling
 - application server instance in cluster 430, 435
 - web application in cluster 432, 437
- disabling deployed applications or modules 332
- discardmanualchanges 82
- display-license command 47, 568
- distributed and local transactions 205
- distributed transactions 249
- DnsBucket
 - monitoring attributes 147
 - monitoring HTTP server elements 143

- DnsBucket attributes 147
 - document directories
 - additional 378
 - primary 356, 378, 629
 - restricting content publication 383
 - document footer, setting 388
 - document preferences
 - default MIME type, specifying a 385
 - directory indexing 384
 - index filenames 384
 - parsing the accept language header 364
 - server home page 385
 - virtual servers, setting 383
 - document root 356, 629
 - setting 378
 - documents
 - lists of those accessed in log 120
 - domain directory 60
 - domain registry
 - recreating 65
 - domains
 - administrative, about 59
 - administrative, creating and deleting if non-root users 62
 - configuring 60
 - creating 61
 - domains.bin 62
 - domains.lck 62
 - dotted names, asadmin 570
 - DTD files
 - application XML 324
 - dynamic deployment 331
 - dynamic redeployment
 - redeploying an existing application without a server restart 186
 - dynamic reloading 332
 - module attributes 587
 - passivation 189
 - pool settings, configuring 197
 - references 227
 - settings, configuring 196
 - types 190
 - EJB container
 - about 188
 - attributes 575
 - attributes that can be monitored 196
 - configuring the log level 194
 - responsibilities 190
 - EJB JAR file 322
 - EJB JAR modules
 - deploying 336
 - ejb-container 101, 575
 - EJBContext 211
 - ejb-jar.xml 228
 - ejb-jar.xml deployment descriptor 324
 - ejb-link element 228
 - ejbLoad 212
 - ejb-module 587
 - monitoring object type 136
 - ejb-name element 228
 - mapping 239
 - EJBObject 190
 - ejb-ref-name element 228
 - eliminateRedundantEndTransaction property 512
 - enable command 568
 - enabled attribute 185, 418, 419
 - enabling
 - application server instance in cluster 430, 432
 - web application in cluster 432
 - Enterprise Java Bean container
 - about 188
 - Enterprise Java Beans
 - entity beans, about 192
 - message-driven beans 193
 - session beans, about 191
 - types 190
 - entity beans
 - about 192
 - bean-managed transactions not allowed 214
 - handling data access via JDBC 190
- ## E
- EJB
 - activation 189
 - cache settings, configuring 197
 - MDB pool settings, configuring 199

- transactions [211](#)
- entity-bean
 - monitoring object type [137](#)
- environment classpath
 - ignore [84](#)
- environment commands, asadmin [553](#)
- environment entries [227](#)
- environment variables
 - AS_ADMIN_PREFIX [561](#)
 - asadmin [597](#)
 - ASADMIN_HOST [555](#)
 - ASADMIN_INSTANCE [555](#)
 - ASADMIN_PASSWORD [555](#)
 - ASADMIN_PORT [555](#)
 - ASADMIN_SECURE [555](#)
 - ASADMIN_USER [555](#)
- error log file [117](#)
- Error qos-error [156](#)
- error responses, customizing [385](#)
- ErrorLogDateFormat [116](#)
- escape characters [512](#)
- escape characters, asadmin [558](#)
- event log file
 - viewing [118](#)
- event variables
 - traps [160](#)
- exceptions
 - rolling back transactions [211](#)
- execution-time-millis [142](#)
- exit status, asadmin [563](#)
- export command [553](#), [568](#)
- external repositories, accessing [237](#)
- external resources
 - about [233](#)
 - creating [235](#)

F

- factory objects [228](#)
- Fail All Connections setting [511](#)
- fail-all-connections property [264](#)
- failover scenarios [603](#)

- FATAL [98](#)
- file cache [344](#)
- file manipulation, remote
 - enabling [379](#)
- FINE [98](#)
- FINER [98](#)
- FINEST [98](#)
- FlagAsyncEnabled [147](#)
- FlagCacheEnabled [147](#)
- FlagEnabled [148](#)
- FlagProfilingEnabled
 - monitoring HTTP server attributes [144](#)
- FlagVirtualServerOverflow
 - monitoring HTTP server attributes [144](#)
- flat transactions, J2EE [206](#)
- flexanlg [548](#)
 - use and syntax [121](#)
- FractionSystemMemoryUsage
 - monitoring Process attributes [145](#)

G

- get command [568](#)
 - asadmin [560](#)
 - monitoring data [131](#)
- getUserTransaction [214](#)
- Guarantee Isolation Level setting [511](#)

H

- HADB [499](#)
- hard links, definition [380](#)
- health checker
 - for cluster [427](#)
- health-checker element [420](#)
- help
 - Administration interface [43](#)
 - asadmin utility [563](#)
- help command [568](#)
- high-availability database [499](#)

- home page 385
- home.html 384
- Hosts 149
- hosts attribute
 - checking against subject pattern 356
- hot deployment
 - deploying an application at server runtime, without restart 186
- htaccess files 380
- HTML, server-parsed, setting up 389
- htpasswd utility 548
- HTTP
 - monitoring 127
- HTTP listeners 352
 - acceptor threads, specifying number 57
 - Administration Server 57
 - attributes 590
 - creating 359
 - http-listener-1 352, 359
 - settings 57
 - SSL/TLS security settings, activating 57
- HTTP server
 - monitable attributes 142
- HTTP server attributes
 - monitoring 143
- HTTP server elements
 - monitoring 142
- HTTP server monitable attributes 142
- HTTP service
 - attributes 581
- http-listener 361, 590
- http-server
 - monitoring attribute names 137
 - monitoring object type 136
- http-server.http-listener 590
- http-service 82, 581
- https-routing property 417

I

- ibm367 387
- ibm819 387
- Id 149
 - monitoring ConnectionQueue attributes 144
 - monitoring HTTP server attributes 144
- idle-timeout-in-seconds 140, 141, 200
- IIOP listener
 - attributes 579
 - creating 317
 - ports 320
 - SSL/TLS settings 319
- IIOP service
 - attributes 577
- IIOP, about 312
- iiop-listener 579
- iiop-service 101, 577
 - monitoring object type 136
- index filename 384
- index.html 384
- in-flight transactions 218
- inflight-tx 139
- INFO 98
 - default log level 97
- INIT 176
- init.conf 82
 - global variable settings at start-up 348
 - termination timeout 74
- initial naming context 230
- initialBeansInPool 196
- inittab 51, 73, 74
 - editing 76
 - restarting servers automatically 76
 - starting the server with 74
- install-license command 568
- instance element 418
- instances
 - application server
 - about 70
 - accessing 46
 - adding to cluster 427, 430
 - disabling in cluster 430, 435
 - enabling in cluster 430, 432
 - quiescing in cluster 435
 - reconfiguring in running cluster 440
 - removing from cluster 442
 - requirements for adding to cluster 429
 - specifying listeners in cluster 431

- specifying quiescing period for 436
 - stopping in cluster 441
 - interactive asadmin 553
 - Interfaces 149
 - internal daemon log rotation 104
 - interval-in-seconds attribute 420
 - IP addresses, in HTTP listeners 352
 - IP-address-based virtual servers 354
 - isFrozen 139
 - iso_646.irv
 - 1991 387
 - iso_8859-1 387
 - 1987 387
 - iso-2022-jp 386
 - iso646-us 387
 - iso-8859-1 386
 - iso-ir-100 387
 - iso-ir-6 387
 - isolation 202
 - iwsCpuID 161
 - iwsCpuIdleTime 161
 - iwsCpuKernelTime 161
 - iwsCpuTable 161
 - iwsCpuUserTime 161
 - iwsInstanceContact 161
 - iwsInstanceCount200 (through 404) 162
 - iwsInstanceCount2xx - 5xx 161
 - iwsInstanceCount3xx 162
 - iwsInstanceCount4xx (& 5xx) 162
 - iwsInstanceCount503 164
 - iwsInstanceCountOther 162
 - iwsInstanceDeathCount 161
 - iwsInstanceDescription 161
 - iwsInstanceId 161
 - iwsInstanceInOctets 161
 - iwsInstanceLoad15MinuteAverage 162
 - iwsInstanceLoad1MinuteAverage 162
 - iwsInstanceLoad5MinuteAverage 162
 - iwsInstanceLocation 161
 - iwsInstanceNetworkInOctets 162
 - iwsInstanceNetworkOutOctets 162
 - iwsInstanceOrganization 161
 - iwsInstanceOutOctets 161
 - iwsInstanceRequests 161
 - iwsInstanceStatus 161
 - iwsInstanceTable 161
 - iwsInstanceUptime 161
 - iwsInstanceVersion 161
 - iwsListenAddress 164
 - iwsListenId 164
 - iwsListenPort 164
 - iwsListenSecurity 164
 - iwsListenTable 164
 - iwsProcessConnectionQueueCount 163
 - iwsProcessConnectionQueueMax 164
 - iwsProcessConnectionQueueOverflows 164
 - iwsProcessConnectionQueuePeak 163
 - iwsProcessConnectionQueueTotal 164
 - iwsProcessId 163
 - iwsProcessKeepaliveCount 164
 - iwsProcessKeepaliveMax 164
 - iwsProcessTable 163
 - iwsProcessThreadCount 163
 - iwsProcessThreadIdle 163
 - iwsThreadPoolTable 164
 - iwsVsCount200 (through 404) 163
 - iwsVsCount2xx - 5xx 163
 - iwsVsCount503 164
 - iwsVsCountOther 163
 - iwsVsId 162
 - iwsVsInOctets 163
 - iwsVsOutOctets 163
 - iwsVsRequests 162
 - iwsVsTable 162
- ## J
- J2EE
 - transactional applications 205
 - transactions 202
 - Web Container, about 181
 - J2EE applications

- EJB specification 284
- JMS, and 284
- message-driven beans, *See* MDB
- resources 221
- services 221
- J2EE Connectors
 - resource manager 204
- J2EE modules
 - definition 322
 - dynamic reloading 332
 - naming 325
 - runtime environment 328
- J2EE object references
 - session persistence for 448
- Java Database Connectivity (JDBC) API
 - for data access via entity beans 190
- Java Message Service, *See* JMS
- Java Virtual Machine, *See* JVM
- java.sql.Connection 214
- java.util.Properties 189
- java-config 577
- JavaMail
 - Folder objects 270
 - JAF 271
 - Message class 269
 - Message subclass 270
 - Session class 271
 - Store class 270
- JavaMail API
 - about 267
 - message handling 268
- JavaMail resources
 - about 267
 - attributes 586
 - configuration parameters 272
- JavaMail Sessions
 - configuring 277
 - creating 275
 - deployment descriptors 274
 - resource factory 228
- javax.ejb.EJBContext 214
- javax.ejb.EntityBean 189
- javax.ejb.EntityContext 189
- javax.ejb.MDBContext 189
- javax.ejb.SessionBean 189
- javax.ejb.SessionContext 189
- javax.ejb.SessionSynchronization 189
- javax.sql.DataSource 205
- javax.sql.XADataSource 205
- JDBC
 - API 190, 222, 246
 - connection factories 228
 - connections 252
 - DataSource object 222
 - datasources 222, 248
 - transactions 265
 - URLs 253
- JDBC connection pools
 - attributes 583
 - connection validation 257, 264
 - creating 254
 - fail-all-connections property 264
 - monitoring 265
 - pool settings 256
 - properties 256
 - transaction isolation 257
- JDBC resources
 - attributes 582
 - creating 250
 - registering 250
- jdbc-connection-pool 261, 583
 - monitoring attribute names 139
 - monitoring object type 136
- jdbc-resource 582
- JMS
 - about 280
 - administered objects *See* JMS administered objects
 - API, list of specifications 193
 - destinations, *See* JMS destinations
 - message consumers 283
 - message delivery models 281
 - message listeners 286
 - message producers 283
 - message structure 282
 - message-driven beans 193
 - messaging system concepts 280
 - physical destinations, *See* JMS destinations
 - programming model 282
 - provider, *See* JMS provider
 - resources, *See* JMS administered objects

- Service, *See* JMS Service
 - specification 280, 282
 - system architecture 281
 - JMS administered objects
 - about 284
 - attributes 585
 - connection factory 291
 - destination 291
 - managing 302
 - JMS destinations 223
 - about 290
 - managing 299
 - queues 290
 - topics 290
 - JMS provider
 - about 279, 287
 - native 279, 294
 - resource manager 204
 - JMS Service
 - administration of 295
 - administration tools 294
 - architecture 292
 - attributes 573
 - built-in 293, 294
 - configuring 296
 - disabling 294
 - external 294
 - MQ administered objects, and 293
 - MQ client runtime, and 293
 - MQ Message Server, and 293
 - jms-max-messages-load 140
 - jms-ping command 309, 568
 - jms-resource 309, 585
 - jms-service 101, 309, 573
 - JNDI
 - architecture 223
 - connection factories 232
 - custom resources, creating 233
 - external repositories 237
 - external resources, creating 235
 - JMS administered objects, and 291
 - lookup method 223
 - lookup names 326
 - lookups 284
 - lookups and associated references 225
 - MDB and 286
 - names 224
 - resource attributes 582
 - JNDI name
 - specifying for session persistence
 - configuration 484
 - JNDI Name setting 513
 - jndi-resource 582
 - JVM
 - attributes 577
 - debug options 74
 - options 85
 - settings
 - configuring 83, 86
 - JVM Profiler
 - attributes 596
 - configuring via Administration interface 85
- ## K
- KeepaliveBucket
 - monitoring HTTP server elements 143
 - keepmanualchanges 82
- ## L
- latin1 387
 - library, shared, using 339
 - license commands 47
 - lifecycle modules
 - attributes 595
 - deploying 337
 - lifecycle-module 595
 - list command 568
 - monitoring 130
 - list-acls command 568
 - list-authdbs command 568
 - list-auth-realms command 568
 - list-components command 568
 - list-custom-resources command 568
 - list-domains command 63, 568

- listen sockets, *See* HTTP listeners
- listener
 - specifying for application server instances in cluster 431
- listener, HTTP
 - editing 57
- listeners attribute 419
- list-file-groups command 568
- list-file-users command 568
- list-http-listeners command 360, 568
- list-iiop-listeners command 568
- list-instances command 568
- list-javamail-resources command 568
- list-jdbc-connection-pools command 261, 569
- list-jdbc-resources command 263, 569
- list-jmsdest command 309, 569
- list-jms-resources command 309, 569
- list-jndi-resources command 569
- list-lifecycle-modules command 337, 569
- list-mimes command 349, 569
- list-persistence-resources command 569
- list-profilers command 569
- list-sub-components command 569
- list-virtual-servers command 569
- load balancer
 - using multiple 443
- load balancing
 - described 395
- Load15MinuteAverage
 - monitoring HTTP server attributes 144
- Load5MinuteAverage
 - monitoring HTTP server attributes 144
- loadbalancer element 415
- loadbalancer.xml file
 - elements in 415
 - location 413
- LoadMinuteAverage
 - monitoring HTTP server attributes 144
- local and distributed transactions 205
- local option 556
- local transaction optimization 205
- location 184
- log analyzer
 - archiving server logs prior to use 121
 - flexanlg, use and syntax 121
 - running 121
 - running from command line 121
- log archive file format 104
- log files
 - access 117
 - archiving 103
 - configuring 120
 - error 117
 - virtual servers 358
- log levels
 - about 96
 - ALERT 97
 - configuring, EJB container 194
 - order of severity 97
 - table of 97
 - used for syslog configuration 98
- log rotation
 - internal daemon 104
 - internal-daemon 104
 - performing (four methods) 104
 - scheduler 105
- log service attributes 580
 - file 114
 - level 114
 - log-stderr 114
 - log-stdout 114
 - use-system-logs 114
- log service element 100
- LOG_ALERT 99
- LOG_CRIT 99
- LOG_DEBUG 99
- LOG_ERR 99
- LOG_INFO 99
- LOG_WARNING 99
- logadm 106
- logadm.conf
 - location and sample of 106
- LogFlushInterval 116
- logging
 - about 90
 - access file, viewing 117
 - Application Client Container (ACC) 102
 - client side 102

- components and subsystems, configuring 115
- components and subsystems, list of 115
- configuring attributes via Administration interface 113
- configuring via administrative interface 111
- configuring via command line interface 110
- directives, configuring 116
- event file, viewing 118
- features and functions 89
- messages
 - information provided 90
- preferences 120
- redirecting application and server log output 103
- UNIX 91
- using syslog 92
- virtual server instances 99
- web container, default behavior 187

log-service 101, 103, 580

log-virtual-server-id 100

long options 597

M

mail-resource 586

managed objects 160, 165

management information base (MIB)

- defines managed objects 160

master agent

- CONFIG file, editing 173
- SNMP 159
- SNMP, enabling and starting 172
- SNMP, installing 167, 169, 172
- SNMP, manually configuring 173
- SNMP, starting 176
- SNMP, starting on another port 172
- starting on a nonstandard port 176

maxBeansInCache 196

max-beans-in-cache 141

maxBeansInPool 196

MaxByteTransmissionRate 150

MaxCacheEntries 147

MaxConnections 147

MaxEntries 148

MaxHeapCacheSize 148

MaxMmapCacheSize 148

MaxOpenConnections 150

MaxOpenEntries 148

max-pool-size 140

MaxProcs

- monitoring HTTP server attributes 144

MaxQueued 147

- monitoring ConnectionQueueBucket attributes 146

MaxThreads 146

- monitoring HTTP server attributes 144

MaxVirtualServers

- monitoring HTTP server attributes 144

MDB

- about 193, 285
- deployment descriptor 286
- JNDI and 286
- transactions 211, 214

MDB container

- about 286
- attributes 574

MDB pool settings

- configuring for EJBs 199

mdb-container 101, 574

message brokers, *See* MQ brokers

message listeners 283, 286

message, log

- information provided 90

message-driven beans, *See* MDB

message-driven-bean

- monitoring object type 137

MessageListener 193

messaging

- asynchronous 279
- JMS *See* JMS

meta-data connection validation 264

metric interval

- time used in traffic calculations 152

MIME types

- attributes 591
- charset parameter 386
- creating new file 349
- definition 349

- definition and accessing page 638
- editing definitions 349
- specifying a default 385
- virtual server settings, configuring 363
- with virtual servers 363
- mime, dotted name 591
- minBeansInCache 196
- minBeansInPool 196
- Mode 149
 - monitoring Process attributes 145
- modified-attribute
 - setting persistence scope to 476
- modified-session
 - setting persistence scope to 473
- monitoring 146, 147, 148, 149
 - about 125
 - additional subsystems and components 127
 - bean-cache attributes 140
 - bean-method attributes 141
 - CacheBucket 143
 - CLI name mapping 132
 - client name mapping, example 133
 - ConnectionQueue 143
 - ConnectionQueue server attributes 144
 - ConnectionQueueBucket 143
 - ConnectionQueueBucket attributes 146
 - ConnectionQueueBucket ConnectionQueue attribute 146
 - ConnectionQueueBucket CountOverFlow attribute 146
 - ConnectionQueueBucket CountQueued attribute 146
 - ConnectionQueueBucket CountTotalConnection attribute 146
 - ConnectionQueueBucket CountTotalQueued attribute 146
 - ConnectionQueueBucket MaxQueued attribute 146
 - ConnectionQueueBucket PeakQueued attribute 146
 - ConnectionQueueBucket TicksTotalQueued attribute 146
 - container subsystems 128
 - DnsBucket 143
 - FlagProfilingEnabled 144
 - FlagVirtualServerOverflow 144
 - HTTP 127
 - HTTP server attributes 142, 143
 - HTTP server elements 142
 - http-server attributes 137
 - Id 144
 - JDBC connection pools 265
 - jdbc-connection-pool attributes 139
 - KeepaliveBucket 143
 - Load15MinuteAverage 144
 - Load5minuteAverage 144
 - LoadMinuteAverage 144
 - MaxProcs 144
 - MaxThreads 144
 - MaxVirtualServers 144
 - Node Process attribute 145
 - object types 135
 - ORB service 129
 - orb-connection attributes 139
 - orb-thread attributes 139
 - Pid Process attribute 145
 - process 143
 - process attributes 138, 145
 - Process CountConfigurations attribute 145
 - Process FractionSystemMemoryUsage attribute 145
 - Process SizeResident attribute 145
 - Process SizeVirtual attribute 145
 - Profile 143
 - Profile attributes 145
 - ProfileBucket 143
 - quality of service (QOS) 129
 - RateBytesReceived 144
 - RateBytesTransmitted 144
 - RequestBucket 143
 - SecondsRunning 144
 - server 142
 - SNMP 126
 - statistics 126
 - Thread 143
 - ThreadPool 143
 - ThreadPool attributes 145
 - ThreadPoolBucket 143
 - TicksPerSecond 144
 - TimeStarted 144
 - TimeStarted Process attribute 145
 - transaction service 129
 - transaction-service attributes 139

- using asadmin to extract data [130](#)
- using get command [131](#)
- using list command [130](#)
- VersionServer [144](#)
- VirtualServer [143](#)
- virtual-server attributes [138](#)

MQ

- about
- administered objects [291](#)
- administration tools [291](#)
- brokers [288](#)
- client runtime [290](#)
- integration with Sun ONE Application Server [292](#)
- message server [288](#)
- messaging system, parts of [287](#)
- purchasing MQ Enterprise Edition [287](#), [616](#)
- resource manager [204](#)
- running trial version of MQ Enterprise Edition [287](#), [615](#)

- multimode [552](#), [569](#)

N

- name attribute [418](#)
 - for property element [416](#)
- naming
 - COSNaming [230](#)
 - initial context [230](#)
 - J2EE modules [325](#)
 - JNDI and resource reference [225](#)
 - JNDI lookup [326](#)
 - services [254](#)
 - standards [325](#)
- native SNMP daemon
 - restarting [169](#)
- network management station (NMS) [158](#)
 - about [159](#)
- nice [365](#)
- non-interactive asadmin [553](#)
- nsfc.conf
 - file cache settings [344](#)
- numBeansCreated [196](#)

- numBeansDestroyed [196](#)
- numBeansInPool [196](#)
- num-beans-in-pool [140](#)
- num-expired-sessions-removed [141](#)
- num-passivation-errors [141](#)
- num-passivations [141](#)
- num-passivation-success [141](#)
- numThreadsWaiting [196](#)
- num-threads-waiting [140](#)

O

- obj.conf file [82](#)
 - set up SAFs for using quality of service [153](#)
 - template [355](#)
 - virtual server [355](#)
- object types, monitoring [135](#)
- online help
 - Administration interface, accessing [43](#)
 - asadmin utility [563](#)
- online upgrade
 - using multiple clusters for [439](#)
- onMessage [137](#), [211](#)
- operands, asadmin [551](#)
- optimization, local transaction [205](#)
- options [550](#)
 - boolean [551](#)
 - default values [597](#)
- optionsfile [455](#)
- ORB
 - attributes [577](#)
 - configuring [314](#)
 - functionality of bundled [313](#)
 - IIO listener configuration [317](#)
 - introduction [312](#)
 - listener attributes [579](#)
 - service, monitoring [129](#)
 - thread pools [316](#)
- orb
 - dotted name [577](#)
 - monitoring object type [136](#)
- orb-connection

- monitoring attribute names [139](#)
 - monitoring object type [136](#)
- orblister [579](#)
- orb-thread-pool
 - monitoring attribute names [139](#)
 - monitoring object type [136](#)

P

- package-appclient utility [548](#)
- passivation [189](#)
- password file option [555](#)
- password file, loading on startup [383](#)
- password property [512](#)
- password.conf [71](#)
- PeakQueued [147](#)
 - monitoring ConnectionQueueBucket attributes [146](#)
- performance
 - dynamic reloading [332](#)
 - using quality of service (QOS) [129](#)
- persistence
 - about [240](#)
 - bean-managed [192](#)
 - container-managed [192](#)
 - data store and [240](#)
 - entity beans [241](#)
- persistence frequency [469](#)
 - comparison of options [472](#)
 - time-based [470](#)
 - web-method [469](#)
- persistence manager
 - creating [243](#)
 - factory resource attributes [585](#)
 - role of [241](#)
- persistence scope [472](#)
 - comparison of options [478](#)
 - modified-attribute [476](#)
 - modified-session [473](#)
 - session [474](#)
- persistence type
 - comparison [468](#)
 - setting [462](#)

- persistence-manager-factory-resource [585](#)
- Pid
 - monitoring Process attributes [145](#)
- PidLog [116](#)
- pipe, with asadmin [558](#)
- pkgadd [47](#)
- Pool Name setting [513](#)
- pool settings
 - EJB, configuring [197](#)
- PooledConnection object [249](#)
- pool-resize-quantity [140](#)
- ports
 - HTTP listener [353](#)
 - IIOP listener [320](#)
- PR_Recv()/net_read [157](#)
- PR_Send()/net_write [157](#)
- PR_TransmitFile [157](#)
- preferences, log
 - setting [120](#)
- primary document directory, setting [356](#), [378](#), [629](#)
- process
 - attribute [142](#)
 - monitoring attribute names [138](#)
 - monitoring attributes [145](#)
 - monitoring HTTP server elements [143](#)
 - monitoring object type [136](#)
- Process element [142](#)
- Profile [151](#)
 - monitoring attributes [145](#)
 - monitoring HTTP server elements [143](#)
- Profile element [142](#)
- ProfileBucket
 - monitoring HTTP server elements [143](#)
- ProfileBucket element [142](#)
- profiler [86](#)
 - attributes [596](#)
 - dotted name [596](#)
- programming, JMS programming model [282](#)
- properties, about [416](#)
- property element [416](#)
- protocol data units (PDUs) [165](#)
- proxy agent, SNMP [167](#)
 - installing [168](#)
 - starting [169](#)

public directories
 configuring 381

Q

qos-error, Error 156
 qos-handler, AuthTrans 156
 quality of service 152
 concurrent connections, virtual servers 158
 configuring 153
 configuring for HTTP server 345
 example 152
 monitoring 129
 only HTTP bandwidth for application level
 measured 156
 set up SAFs in obj.conf for using 153
 using 129
 virtual servers, configuring settings for 365
 queues, *See* JMS destinations
 quiescing 434
 application server instance in cluster 435
 web application in cluster 437
 quiescing period 435
 modifying while quiescing is on 438
 specifying 430
 specifying for application server instances 436
 specifying for web application 433

R

ra.xml deployment descriptor 324
 RAR files 322
 RateBytesReceived
 monitoring HTTP server attributes 144
 RateBytesTransmitted 150
 monitoring HTTP server attributes 144
 rc.2.d, starting the server with 74
 recompute interval 152
 reconfig command 66, 81, 261, 560, 569
 Recovery, Transaction 206
 redeploying applications 331

registry, domains
 recreating 65
 reloading, dynamic 332
 reload-poll-interval-in-seconds property 417
 remote file manipulation
 enabling 379
 RemoteException 210
 removal-timeout-in-seconds 199
 request processing, for virtual servers 356
 RequestBucket
 monitoring HTTP server elements 143
 requirement rules 414
 require-monitor-data property 417
 Resource Adapter 203
 resource environment references 229, 238
 resource manager
 database 203
 definition 203
 J2EE Connectors 204
 JMS provider 204
 transaction 204
 Resource RAR files 322
 resource references 225, 237
 resources
 custom 233
 external 233
 JMS, *See* JMS administered objects
 monitoring object type 136
 response-timeout-in-seconds property 417
 res-sharing-scope 207
 restart-instance command 77, 569
 restartserv 78
 restricting symbolic links 380
 RMI
 introduction 313
 RMI/IIOP clients
 deploying 338
 rollbacks, *See* transactions, rollbacks
 root
 monitoring object type 136
 runtime environments 328

S

- sagt 168
- sagt, command for starting Proxy SNMP agent 169
- scheduler log rotation
 - archive log files 106
 - scheduler link 105
- schedulerd 106
- script, asadmin 558
- SecondsMaxAge 148
- SecondsRunning
 - monitoring HTTP server attributes 144
- SecondsTimeouts 148
- security service
 - attributes 580
- security, asadmin 564
- security-service 101, 580
- Server element 142
- server instance
 - adding 79
 - deleting 80
- server logs 121
- server.log 90
 - default log level 97
 - default logging 91
 - example 91
- server.xml 82, 95, 100, 103, 215, 331, 343, 352
 - default web application 185
 - settings that do not require a restart 82
- server1 70
- serverList property 512
- server-parsed HTML 389
- servers
 - configuration attributes 596
 - monitoring HTTP server elements 142
 - restarting (Unix) 74
 - restarting manually (Unix) 51, 73
 - starting 74
 - stopping 53
 - stopping manually 53
 - stopping manually (Unix) 73
- session
 - setting persistence scope to 474
- session beans
 - about 191
 - instance variables, synchronizing 213
 - stateful 191
 - stateless 191
 - synchronizing instance variables 213
 - transactions 211, 212
- session persistence 445
 - about 446
 - basic configuration steps 450
 - configuring 459
 - configuring other properties 478
 - for J2EE object references 448
 - specifying cluster name 482
 - specifying JNDI name 484
- session store
 - creating 451
 - specifying options in file 455
- sessions
 - and dynamic reloading 331, 332
 - JMS messaging 283
- SessionSynchronization 212, 213
- set command 560, 569
- setAutoCommit 214
- setRollbackOnly 211
- settings
 - Administration Server, accessing 54
 - Java Virtual Machine (JVM), configuring 83
- SEVERE 98
- shared library, using 339
- short options 597
- show-component-status command 569
- show-instance-status command 83, 569
- shutdown command 53, 569
- Simple Network Management Protocol (SNMP)
 - introduction 158
- single sign-on
 - availability of session state 457
- single sign-on, about 186
- singlemode 552
- SizeHeapCache 148
- SizeMmapCache 148
- SizeResident
 - monitoring Process attributes 145
- SizeVirtual
 - monitoring Process attributes 145

SMUX 167

SNMP

- community string 166
- community strings, configuring 166
- daemon, restarting 169
- GET and SET messages 164
- master agent 159
 - enabling and starting 172
 - installing 167, 169, 172
 - manually configuring 173
 - starting 176
 - starting on another port 172
- monitoring 126
- native daemon, restarting 169
- proxy agent
 - about 167
 - installing 168
 - starting 169
- setting up on a server 166
- Simple Network Management Protocol,
 - introduction 158
- subagent 159
- subagent, enabling 178
- trap 165

snmpd, command for restarting native SNMP daemon 169

soft (symbolic) links 380

Solaris 9 bundled installations
configuring 37

specify

- log file 116
- log level 115
- transaction log location 116

SSL/TLS

- HTTP listener settings 57
- IIO listener settings 319
- using with virtual servers 357

standalone-ejb-module
monitoring object type 137

start-appserv command 569

start-domain command 51, 64, 569

starting the server 74

start-instance command 72, 74, 569

startserv 72

starting the Administration Server 51

state, virtual server 364

stateful-session-bean
monitoring object type 137

stateless-session-bean
monitoring object type 137

statistics

- monitoring 126
- quality of service bandwidth lost when server
reconfigured dynamically 157
- settings for measuring traffic 152

status, application server instance 83

stderr 91, 103

stdout 91, 103

Steady Pool Size setting 511

steady-pool-size 140

stop-appserv command 53, 569

stop-domain command 53, 64, 569

stop-instance command 72, 570

stopping the server 53

stopserv 72

shutting down the Administration Server 53

storepassword 454

storeuser 454

stronger ciphers 391

subagent

SNMP 159

SNMP, enabling 178

subelements, about 414

subsystem

logging control, at the 101

logging default handlers 101

summary

monitorable attribute 138

Sun customer support 31

Sun ONE Message Queue, *See* MQ

Sun ONE Studio

about 46

deploying with 335

sun-acc.xml 102

sun-application.xml deployment descriptor 325

sun-application-client.xml deployment
descriptor 325

sun-cmp-mapping.xml deployment descriptor 325

- sun-ejb-jar.xml deployment descriptor 325
- sun-loadbalancer_1_0.dtd file 413
- sun-web.xml 184
- sun-web.xml deployment descriptor 325
- symbolic (soft) links, definition 380
- symbolic links, restricting 380
- syntax, asadmin 550
- sysContact 173, 174
- sysLocation 173, 174
- syslog
 - info used to identify Application Server messages 95
 - log levels used for configuration 98
 - logging 92
 - message
 - example 96
- syslog.conf 93
 - configuring to store less severe messages 93
 - example of configured file 93
- syslogd 93
- system RC scripts
 - restarting the server automatically 76
- System.getCurrentTimeInMillis 217

T

- table connection validation 264
- Table Name setting 511
- termination timeout
 - init.conf 74
 - setting 74
- Thread
 - monitoring attributes 149
 - monitoring HTTP server elements 143
- thread pools
 - information you specify to add 347
 - ORB 316
- ThreadPool
 - monitoring attributes 145
 - monitoring HTTP server elements 143
- Thread-pool 146
- ThreadPoolBucket
 - monitoring attributes 146
 - monitoring HTTP server elements 143
- thread-pool-size 139
- three-tier database access 247
- TicksDispatch 151
- TicksFunction 151
- TicksPerSecond
 - monitoring HTTP server attributes 144
- TicksTotalQueued
 - monitoring ConnectionQueueBucket attributes 146
- time interval for health check
 - in cluster 427
- time-based
 - setting persistence frequency to 470
- timeout value for health check
 - in cluster 427
- timeout, termination
 - setting 74
- timeout-in-seconds attribute 420
- timeStamp 217
- TimeStarted 149
 - monitoring HTTP server attributes 144
 - monitoring Process attributes 145
- TLS Rollback option
 - ciphers 319
- tools
 - available for administration functions 36
- topics *See* JMS destinations
- total-beans-created 140
- total-beans-destroyed 140
- total-beans-in-cache 141
- total-connections-timed-out 139
- total-inbound-connections 139
- total-num-calls 141
- total-num-errors 141
- total-num-success 142
- total-outbound-connections 139
- total-threads-waiting 139
- total-tx-completed 139
- total-tx-inflight 139
- total-tx-rolled-back 139
- traffic

- settings, counting statistics for [152](#)
- Transaction Isolation setting [511](#)
- Transaction Manager [202](#)
- transaction resource manager [204](#)
- transaction service
 - administering via asadmin [151](#)
 - attributes [574](#)
 - freezing and unfreezing examples [219](#)
 - monitoring [129](#)
- transactional User Application [203](#)
- TransactionRequiredException [209](#)
- transactions
 - administering with Administration interface [215](#)
 - afterBegin method example [213](#)
 - afterCompletion method example [213](#)
 - attributes [208](#), [646](#)
 - bean-managed [195](#)
 - commits [195](#)
 - consistency [202](#)
 - container-managed [207](#)
 - databases, administering and monitoring using
 - asadmin [218](#)
 - distributed [249](#)
 - entity beans [211](#)
 - flat, J2EE [206](#)
 - in-flight [218](#)
 - introduction [201](#)
 - J2EE [202](#)
 - local and distributed [205](#)
 - local optimization [205](#)
 - Mandatory attribute [209](#)
 - message-driven beans [211](#), [214](#)
 - monitoring [219](#)
 - Never attribute [210](#)
 - NotSupported attribute [210](#)
 - recovery [206](#)
 - required attribute [209](#)
 - RequiresNew attribute [209](#)
 - rollbacks [195](#), [211](#), [219](#)
 - session beans [212](#)
 - Supports attribute [210](#)
 - user application [203](#)
- transactionsCompleted [216](#)
- transaction-service [101](#), [574](#)
 - monitoring attribute names [139](#)
 - monitoring object type [136](#)

- transactionsInFlight [217](#)
- transactionsRecovered [216](#)
- transactionsRolledBack [216](#)
- traps
 - messages containing event variables [160](#)
 - SNMP [165](#)
- two-tier database access [247](#)

U

- ulimit [71](#)
- undeploy command [334](#), [570](#)
- unset command [554](#), [570](#)
- update-file-user command [570](#)
- url attribute [420](#)
- URL connection factory resources [238](#)
- URL forwarding, configuring [388](#)
- URL-host-based virtual servers [354](#)
- URLs, JDBC [253](#)
- us [387](#)
- usage, asadmin [564](#)
- us-ascii [386](#)
- User Application, transactional [203](#)
- user directories
 - configuring [381](#)
 - customizing [381](#)
- user transaction references [230](#)
- username property [512](#)
- UserTransaction object [230](#)
- use-system-logging [93](#)

V

- Validation Method setting [511](#)
- value attribute [416](#)
- variables
 - event
 - traps [160](#)
 - global

- settings in init.conf 348
- verifier utility 323, 548
- version command 570
- VersionServer
 - monitoring HTTP server attributes 144
- viewing events 123
- virtual servers
 - acceptor threads 353
 - attributes 184, 593
 - concurrent connections, quality of service 158
 - configuring MIME settings 363
 - configuring web containers to deploy web applications 183
 - creating 362
 - default 355
 - default configuration example 369
 - default web application 185
 - deleting 368
 - deploying 369
 - document preferences, setting 383
 - editing general settings 366
 - HTTP listeners 352
 - HTTP listeners, creating 359
 - intranet hosting example 372
 - introduction 351
 - log files 358
 - logging instances 99
 - mass hosting example 374
 - public directories, configuring to use 381
 - quality of service, configuring settings 365
 - request processing 356
 - secure server example 370
 - setting additional document directories 378
 - single sign-on 186
 - state 364
 - types 354
 - using access control 358
 - using quality of service 129
 - using SSL 357
- VirtualServer
 - monitoring attributes 149
 - monitoring HTTP server elements 143
- virtual-server 367, 593
 - monitoring attribute names 138
 - monitoring object type 136
- virtual-server attribute 142

- VirtualServer element 142

W

- waiting-thread-count 139
- WAR files 322, 365
- WAR modules, deploying 336
- WARNING 98
- watchdog 78
- web application
 - deploying on cluster 432
 - disabling in cluster 432, 437
 - enabling in cluster 432
 - quiescing in cluster 437
 - redeploying on cluster 441
 - specifying quiescing period for 433
 - undeploying from cluster 440
- web applications 322
 - elements of 183
 - with virtual servers 365
- web container
 - attributes 576
 - default logging behavior 187
 - deploying web applications within virtual servers 183
 - introduction 181
 - web application deployment 185
- web module attributes 589
- web.xml deployment descriptor 324
- web-container 82, 101, 576
- WEB-INF directory 184
- web-method
 - setting persistence frequency to 469
- web-module 589
 - attributes 184
- web-module element 419
- wscompile utility 549
- wsdeploy utility 549

X

XATransaction mode [204](#)

x-euc-jp [387](#)

x-mac-roman [387](#)

x-sjis [386](#)