



Sun Java System Application Server Platform Edition 9 Administration Guide



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

Part No: 819-3658-11
December 2008

Copyright 2008 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

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.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and SunTM Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

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

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

Copyright 2008 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plusieurs brevets américains ou des applications de brevet en attente aux Etats-Unis et dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certains composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays; elle est licenciée exclusivement par X/Open Company, Ltd.

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

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

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

Contents

Preface	21
1 Getting Started	29
About the Sun Java System Application Server	29
What is the Application Server?	29
Application Server Architecture	31
Tools for Administration	34
About the Administration Console	35
Using Online Help	36
Configuring the Application Server	36
Creating a Domain	36
Deleting a Domain	37
Listing Domains	37
Starting the Domain	37
Stopping the Domain	37
Configuration Changes	38
Changing Application Server Configuration	38
Ports in the Application Server	38
Further Information	39
2 JDBC Resources	41
About JDBC Resources	41
Creating a JDBC Resource	41
About JDBC Connection Pools	42
Creating a JDBC Connection Pool	42
How JDBC Resources and Connection Pools Work Together	43
Configurations for Specific JDBC Drivers	43

Java DB Type 4 Driver	44
Sun Java System JDBC Driver for DB2 Databases	45
Sun Java System JDBC Driver for Oracle 9i and 10g Databases	45
Sun Java System JDBC Driver for Microsoft SQL Server Databases	46
Sun Java System JDBC Driver for Sybase Databases	46
IBM DB2 8.1 and 8.2 Type 2 Driver	47
Oracle Thin Type 4 Driver for Oracle 9i and 10g Databases	47
Microsoft SQL Server JDBC Driver	48
PostgreSQL JDBC Driver	49
MM MySQL Type 4 Driver (Non-XA)	49
MM MySQL Type 4 Driver (XA Only)	50
JConnect Type 4 Driver for Sybase ASE 12.5 and 15 Databases	51
Inet Oraxo JDBC Driver for Oracle 9i and 10g Databases	51
Inet Merlia JDBC Driver for Microsoft SQL Server Databases	52
Inet Sybelux JDBC Driver for Sybase Databases	53
OCI Oracle Type 2 Driver for Oracle 9i and 10g Databases	53
IBM Informix Type 4 Driver	54
3 Configuring Java Message Service Resources	55
About JMS Resources	55
The JMS Provider in the Application Server	55
JMS Resources	55
The Relationship Between JMS Resources and Connector Resources	57
JMS Connection Factories	57
Equivalent asadmin command	57
JMS Destination Resources	58
Equivalent asadmin commands	58
JMS Physical Destinations	58
Equivalent asadmin commands	58
JMS Providers	59
Configuring General Properties for the JMS Provider	59
Foreign JMS Providers	60
Configuring the Generic Resource Adapter for JMS	60
▼ To Configure the Generic Resource Adapter	60
Resource Adapter Properties	61

ManagedConnectionFactory Properties	66
Administered Object Resource Properties	66
Activation Spec Properties	67
4 Configuring JavaMail Resources	71
Creating a JavaMail Session	71
5 JNDI Resources	73
Java EE Naming Services	73
Naming References and Binding Information	74
Using Custom Resources	75
Using External JNDI Repositories and Resources	75
6 Connectors	77
Connector Connection Pools	77
Connector Resources	79
Administered Object Resources	79
7 Java EE Containers	81
About the Java EE Containers	81
Types of Java EE Containers	81
The Web Container	81
The EJB Container	82
Configuring Java EE 5 Containers	82
Configuring Web Container Sessions	83
Configuring the General EJB Settings	85
Configuring the Message-Driven Bean Settings	87
Configuring the EJB Timer Service Settings	87
8 Configuring Security	89
About Application Server Security	89
Overview of Security	89
About Authentication and Authorization	94
Understanding Users, Groups, Roles, and Realms	97

Introduction to Certificates and SSL	100
About Firewalls	103
Managing Security With the Admin Console	103
Configuring Security	106
Configuring General Security Settings	106
Granting Access to Administration Tools	108
Configuring Realms	108
Creating a Realm	108
Editing a Realm	109
Setting the Default Realm	110
Additional Information for Specific Realms	110
Configuring JACC Providers	116
Creating a JACC Provider	116
Setting the Active JACC Provider	117
Configuring Audit Modules	117
Creating an Audit Module	117
Enabling or Disabling Audit Logging	118
Setting the Active Audit Module	118
Configuring Listeners and JMX Connectors	118
Configuring Security for HTTP Listeners	118
Configuring Security for IIOP Listeners	118
Configuring Security For The Admin Service's JMX Connector	119
▼ To set listener security properties	119
▼ To secure CORBA objects	120
Configuring Security for Virtual Servers	121
▼ To configure single sign-on (SSO)	121
Configuring Connector Connection Pools	123
About Connector Connection Pools	123
About Security Maps	123
Working with Certificates and SSL	127
About Certificate Files	127
Using Java Secure Socket Extension (JSSE) Tools	128
Further Information	131

9	Configuring Message Security	133
	About Message Security	133
	Overview of Message Security	133
	Understanding Message Security in the Application Server	134
	Securing a Web Service	138
	Securing the Sample Application	139
	Configuring the Application Server for Message Security	140
	Admin Console Tasks for Message Security	143
	▼ To enable providers for message security	144
	▼ To configure a message security provider	145
	▼ Creating a Message Security Provider	148
	▼ To create an httpServlet provider	150
	▼ To delete a message security configuration	151
	▼ To delete a message security provider	151
	▼ To enable message security for application clients	151
	Setting the Request and Response Policy for the Application Client Configuration	152
	Further Information	153
10	Transactions	155
	About Transactions	155
	What Is a Transaction?	155
	Transactions in Java EE Technology	156
	Workarounds for Specific Databases	156
	Configuring Transaction Management	157
	Configuring Application Server to Recover Transactions	157
	Setting a Transaction Timeout Value	158
	Specifying the Transaction Logs	159
	Setting the Keypoint Interval	159
11	Configuring the HTTP Service	161
	Virtual Servers	161
	HTTP Listeners	162

12	Managing Web Services	165
	Overview of Web Services	166
	Web Services Standards	166
	Java EE Web Service Standards	167
	Deploying and Testing Web Services	168
	Deploying Web Services	168
	Viewing Deployed Web Services	168
	Testing Web Services	169
	Web Services Security	169
	Publishing to Web Services Registries	170
	Adding a Connector Module for a Registry	170
	Setting up a Registry for use with Application Server	170
	Adding a Registry	173
	Publishing a Web Service to a Registry	173
	Monitoring Web Services	174
	Viewing Web Service Statistics	174
	Viewing Web Service Messages	175
	Transforming Messages with XSLT Filters	175
	Web Services as JBI Service Providers	176
13	Configuring the Object Request Broker	177
	About the Object Request Broker	177
	CORBA	177
	What is the ORB?	178
	IIOP Listeners	178
	Configuring the ORB	178
	Working with IIOP Listeners	178
14	Thread Pools	179
	About Thread Pools	179
	Configuring Thread Pools	180
	Creating a Thread Pool	180
	Editing the Thread Pool Settings	181
	Deleting a Thread Pool	181

15	Configuring Logging	183
	About Logging	183
	Log Records	183
	The Logger Namespace Hierarchy	184
	Configuring Logging	186
	Configuring General Logging Settings	186
	Configuring Log Levels	186
	Viewing Server Logs	187
16	Monitoring Components and Services	191
	About Monitoring	191
	Monitoring in the Application Server	191
	Overview of Monitoring	192
	About the Tree Structure of Monitorable Objects	192
	About Statistics for Monitored Components and Services	196
	Enabling and Disabling Monitoring	212
	To configure monitoring levels using the Admin Console	212
	▼ To configure monitoring using the asadmin tool	213
	Viewing Monitoring Data	214
	Viewing monitoring data in the Admin Console	214
	Viewing Monitoring Data With the asadmin Tool	214
	Using JConsole	229
	Securing JConsole to Application Server Connection	230
	Prerequisites for Connecting JConsole to Application Server	231
	▼ Connecting JConsole to Application Server	231
	▼ Connecting JConsole Securely to Application Server	232
17	Configuring Management Rules	235
	About Management Rules	235
	Admin Console Tasks for Management Rules	236
	Equivalent asadmin Command	240
18	Configuring the Diagnostic Service	241
	What is the Diagnostic Framework?	241

Diagnostic Service Framework	241
Generating a Diagnostic Report	242
19 Java Virtual Machine and Advanced Settings	243
Tuning the JVM Settings	243
Configuring Advanced Settings	244
20 Automatically Restarting a Domain	247
Restarting Automatically on UNIX Platforms	247
Restarting Automatically on the Microsoft Windows Platform	248
Security for Automatic Restarts	249
Executing Local Admin CLI Commands without Admin Password	250
▼ To create the password alias	250
21 Dotted Name Attributes for domain.xml	253
Top Level Elements	253
Elements Not Aliased	255
22 The asadmin Utility	257
The asadmin Command Usage	258
Multi and Interactive Modes	258
Local Commands	259
Remote Commands	259
The Password File	261
Multimode Command	261
The List, Get and Set Commands	261
Server Lifecycle Commands	263
List and Status Commands	264
Deployment Commands	264
Message Queue Administration Commands	265
Resource Management Commands	266
Application Server Configuration Commands	268
General Configuration Commands	268
HTTP, IIOP and SSL Listener Commands	268

Lifecycle and Audit Module Commands	269
Profiler and JVM Options Commands	270
Virtual Server Commands	270
Threadpool Commands	270
Transaction and Timer Commands	271
User Management Commands	271
Monitoring Data Commands	272
Rule Commands	272
Database Commands	273
Diagnostic and Logging Commands	273
Web Service Commands	274
Security Service Commands	274
Password Commands	275
Verify domain.xml Command	276
Custom MBean Commands	276
Miscellaneous Commands	277
 Index	 279

Figures

FIGURE 1-1	Application Server Architecture	32
FIGURE 8-1	Role Mapping	98

Tables

TABLE 1-1	Application Server Listeners that Use Ports	39
TABLE 5-1	JNDI Lookups and Their Associated References	75
TABLE 8-1	Application Server Authentication Methods	95
TABLE 8-2	Required properties for ldap realm	110
TABLE 8-3	Optional properties for ldap realm	111
TABLE 8-4	Example ldap realm values	112
TABLE 8-5	Optional properties for certificate realm	114
TABLE 8-6	Required properties for file realms	114
TABLE 9-1	Message protection policy to WS-Security SOAP message security operation mapping	140
TABLE 15-1	Application Server Logger Namespaces	184
TABLE 16-1	EJB Statistics	196
TABLE 16-2	EJB Method Statistics	197
TABLE 16-3	EJB Session Store Statistics	198
TABLE 16-4	EJB Pool Statistics	199
TABLE 16-5	EJB Cache Statistics	199
TABLE 16-6	Timer Statistics	200
TABLE 16-7	Web Container (Servlet) Statistics	200
TABLE 16-8	Web Container (Web Module) Statistics	201
TABLE 16-9	HTTP Service Statistics (applicable to Platform Edition only)	202
TABLE 16-10	JDBC Connection Pool Statistics	203
TABLE 16-11	Connector Connection Pool Statistics	205
TABLE 16-12	Connector Work Management Statistics	206
TABLE 16-13	Connection Manager (in an ORB) Statistics	206
TABLE 16-14	Thread Pool Statistics	206
TABLE 16-15	Transaction Service Statistics	207
TABLE 16-16	JVM Statistics	208
TABLE 16-17	JVM Statistics for J2SE 5.0 - Class Loading	208
TABLE 16-18	JVM Statistics for J2SE 5.0 - Compilation	209

TABLE 16-19	JVM Statistics for J2SE 5.0 - Garbage Collection	209
TABLE 16-20	JVM Statistics for J2SE 5.0 - Memory	209
TABLE 16-21	JVM Statistics for J2SE 5.0 - Operating System	210
TABLE 16-22	JVM Statistics for J2SE 5.0 - Runtime	210
TABLE 16-23	JVM Statistics for J2SE 5.0 - Thread Info	211
TABLE 16-24	JVM Statistics for J2SE 5.0 - Threads	212
TABLE 16-25	Top Level	223
TABLE 16-26	Applications Level	223
TABLE 16-27	Applications - Enterprise Applications and Standalone Modules	224
TABLE 16-28	HTTP-Service Level	227
TABLE 16-29	Thread-Pools Level	227
TABLE 16-30	Resources Level	228
TABLE 16-31	Transaction-Service Level	228
TABLE 16-32	ORB Level	228
TABLE 16-33	JVM Level	229
TABLE 22-1	Remote Commands Required Options	259
TABLE 22-2	Server Lifecycle Commands	263
TABLE 22-3	List and Status Commands	264
TABLE 22-4	Deployment Commands	265
TABLE 22-5	Message Queue Commands	265
TABLE 22-6	Resource Management Commands	266
TABLE 22-7	General Configuration Commands	268
TABLE 22-8	IIOP Listener Commands	269
TABLE 22-9	Lifecycle Module Commands	269
TABLE 22-10	Profiler and JVM Options Commands	270
TABLE 22-11	Virtual Server Commands	270
TABLE 22-12	Threadpool Commands	271
TABLE 22-13	Transaction Commands	271
TABLE 22-14	User Management Commands	272
TABLE 22-15	Monitoring Data Commands	272
TABLE 22-16	Rules Commands	272
TABLE 22-17	Database Commands	273
TABLE 22-18	Diagnostic and Logging Commands	273
TABLE 22-19	Web Service Commands	274
TABLE 22-20	Security Commands	275
TABLE 22-21	Password Commands	276

TABLE 22-22	Verify domain.xml Command	276
TABLE 22-23	Custom MBean Commands	276
TABLE 22-24	Miscellaneous Commands	277

Examples

EXAMPLE 16-1	Applications Node Tree Structure	193
EXAMPLE 16-2	HTTP Service Schematic (Platform Edition version)	194
EXAMPLE 16-3	HTTP Service Schematic (Enterprise Edition version)	194
EXAMPLE 16-4	Resources Schematic	194
EXAMPLE 16-5	Connector Service Schematic	195
EXAMPLE 16-6	JMS Service Schematic	195
EXAMPLE 16-7	ORB Schematic	195
EXAMPLE 16-8	Thread Pool Schematic	196
EXAMPLE 20-1	for start-domain	251
EXAMPLE 22-1	Syntax Example	258
EXAMPLE 22-2	help Command Example	258
EXAMPLE 22-3	Passwordfile contents	261

Preface

The *Sun Java System Administration Guide* describes how to configure, manage, and deploy the Application Server subsystems and components.

Who Should Use This Book

This book is intended for information technology administrators in production environments. This guide assumes you are familiar with the following topics:

- Basic system administration tasks
- Installing software
- Using Web browsers
- Starting database servers
- Issuing commands in a terminal window

How This Book Is Organized

The Administration Guide is organized as follows.

TABLE P-1 How This Book Is Organized

Chapter	Description
Chapter 1, “Getting Started”	Describes Application Server system administration.
Chapter 2, “JDBC Resources”	Describes JDBC resources (data source) provides applications with a means of connecting to a database.
Chapter 3, “Configuring Java Message Service Resources”	Describes how to configure resources for applications that use the Java Message Service (JMS) API.
Chapter 4, “Configuring JavaMail Resources”	Describes JavaMail Resources, a set of abstract APIs that model a mail system
Chapter 5, “JNDI Resources”	Explains JNDI resources, which are used for accessing different kinds of naming and directory services.
Chapter 6, “Connectors”	Describes Application Server support for connectors.

TABLE P-1 How This Book Is Organized (Continued)

Chapter	Description
Chapter 7, “Java EE Containers”	Describes Application Server support for containers.
Chapter 8, “Configuring Security”	Describes how to configure Application Server security
Chapter 9, “Configuring Message Security”	Describes message security for the Application Server.
Chapter 10, “Transactions”	Provides information on how to configure transactions in the Application Server.
Chapter 11, “Configuring the HTTP Service”	Describes how to deploy web applications and make deployed web applications accessible by HTTP clients.
Chapter 12, “Managing Web Services”	Describes web services management with Application Server.
Chapter 13, “Configuring the Object Request Broker”	Describes how to configure the Object Request Broker (ORB) and IIOP listeners.
Chapter 14, “Thread Pools”	Describes how to create, edit, and delete thread pools in Application Server.
Chapter 15, “Configuring Logging”	Describes how to use the Admin Console to configure logging and view the server log.
Chapter 16, “Monitoring Components and Services”	Contains information about monitoring components using the Application Server Admin Console
Chapter 17, “Configuring Management Rules”	Contains information about setting administration policies to automate routine administration tasks, configure self-tuning of the application server for diverse runtime condition and improve availability by preventing failures.
Chapter 18, “Configuring the Diagnostic Service”	Explains the diagnostic framework and describes how to generate reports.
Chapter 19, “Java Virtual Machine and Advanced Settings”	Explains how to configure JVM settings.
Chapter 20, “Automatically Restarting a Domain”	Describes automatic restart on various platforms including Windows and Linux.
Chapter 21, “Dotted Name Attributes for domain.xml”	Describes the dotted name attributes that can be used to address the MBean and its attributes.
Chapter 22, “The asadmin Utility”	Describes the command-line administration utility known as asadmin.

Application Server Documentation Set

The Application Server documentation set describes deployment planning and system installation. The Uniform Resource Locator (URL) for stand-alone Application Server documentation is <http://docs.sun.com/app/docs/coll/1343.3>. For an introduction to Application Server, refer to the books in the order in which they are listed in the following table.

TABLE P-2 Books in the Application Server Documentation Set

Book Title	Description
<i>Documentation Center</i>	Application Server documentation topics organized by task and subject.
<i>Release Notes</i>	Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of the supported hardware, operating system, Java™ Development Kit (JDK™), and database drivers.
<i>Quick Start Guide</i>	How to get started with the Application Server product.
<i>Installation Guide</i>	Installing the software and its components.
<i>Application Deployment Guide</i>	Deployment of applications and application components to the Application Server. Includes information about deployment descriptors.
<i>Developer's Guide</i>	Creating and implementing Java Platform, Enterprise Edition (Java EE platform) applications intended to run on the Application Server that follow the open Java standards model for Java EE components and APIs. Includes information about developer tools, security, debugging, and creating lifecycle modules.
<i>Java EE 5 Tutorial</i>	Using Java EE 5 platform technologies and APIs to develop Java EE applications.
<i>Administration Guide</i>	Configuring, managing, and deploying Application Server subsystems and components from the Admin Console.
<i>Administration Reference</i>	Editing the Application Server configuration file, <code>domain.xml</code> .
<i>Upgrade and Migration Guide</i>	Migrating your applications to the new Application Server programming model, specifically from Application Server 6.x, and 7.x, and 8.x. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
<i>Troubleshooting Guide</i>	Solving Application Server problems.
<i>Error Message Reference</i>	Solving Application Server error messages.
<i>Reference Manual</i>	Utility commands available with the Application Server; written in man page style. Includes the <code>asadmin</code> command line interface.

Related Books

- Message Queue documentation: <http://docs.sun.com/app/docs/coll/1307.4>
- Directory Server documentation: <http://docs.sun.com/app/docs/coll/1224.3>
- Web Server documentation: <http://docs.sun.com/app/docs/coll/1653.1>

The URL for all documentation about Sun Java Enterprise System (Java ES) and its components is <http://docs.sun.com/prod/entsys.06q3>.

You can find a directory of URLs for the official specifications at `install-dir/docs/index.htm`. Additionally, the following resources might be useful.

General Java EE Online Information:

The [Java EE 5 Tutorial](http://java.sun.com/javaee/5/docs/tutorial/doc/index.html) (<http://java.sun.com/javaee/5/docs/tutorial/doc/index.html>)

The [Java EE Blueprints](http://java.sun.com/reference/blueprints/index.html) (<http://java.sun.com/reference/blueprints/index.html>)

General Java EE Books:

Core J2EE Patterns: Best Practices and Design Strategies by Deepak Alur, John Crupi, & Dan Malks, Prentice Hall Publishing

Java Security, by Scott Oaks, O'Reilly Publishing

Books on Programming with servlets and JavaServer Pages™ (JSP™) files:

Java Servlet Programming, by Jason Hunter, O'Reilly Publishing

Java Threads, 2nd Edition, by Scott Oaks & Henry Wong, O'Reilly Publishing

Books on Programming with EJB components:

Enterprise JavaBeans, by Richard Monson-Haefel, O'Reilly Publishing

Books on Programming with JDBC:

Database Programming with JDBC and Java, by George Reese, O'Reilly Publishing

JDBC Database Access With Java: A Tutorial and Annotated Reference (Java Series), by Graham Hamilton, Rick Cattell, & Maydene Fisher

Javadoc™ Tool:

A Javadoc tool reference for packages provided with the Application Server is located in *install-dir/docs/api*.

Default Paths and File Names

The following table describes the default paths and file names that are used in this book.

TABLE P-3 Default Paths and File Names

Placeholder	Description	Default Value
<i>install-dir</i>	Represents the base installation directory for Application Server.	Solaris™ and Linux operating system installations, non-root user: <i>user's-home-directory/SUNWappserver</i> Solaris and Linux installations, root user: <i>/opt/SUNWappserver</i> Windows, all installations: <i>SystemDrive:\Sun\AppServer</i>
<i>domain-root-dir</i>	Represents the directory containing all domains.	<i>install-dir/domains/</i>
<i>domain-dir</i>	Represents the directory for a domain. In configuration files, you might see <i>domain-dir</i> represented as follows: <code>\${com.sun.aas.instanceRoot}</code>	<i>domain-root-dir/domain-dir</i>

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P-4 Typographic Conventions

Typeface	Meaning	Example
<i>AaBbCc123</i>	The names of commands, files, and directories, and onscreen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name% you have mail.</code>
AaBbCc123	What you type, contrasted with onscreen computer output	<code>machine_name% su</code> Password:
<i>AaBbCc123</i>	A placeholder to be replaced with a real name or value	The command to remove a file is <i>rm filename</i> .
<i>AaBbCc123</i>	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the <i>User's Guide</i> . A <i>cache</i> is a copy that is stored locally. Do <i>not</i> save the file.

Symbol Conventions

The following table explains symbols that might be used in this book.

TABLE P-5 Symbol Conventions

Symbol	Description	Example	Meaning
[]	Contains optional arguments and command options.	ls [-l]	The -l option is not required.
{ }	Contains a set of choices for a required command option.	-d {y n}	The -d option requires that you use either the y argument or the n argument.
\${ }	Indicates a variable reference.	\${com.sun.javaRoot}	References the value of the com.sun.javaRoot variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Accessing Sun Resources Online

The docs.sun.comSM web site (<http://docs.sun.com>) enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. Books are available as online files in PDF and HTML formats. Both formats are readable by assistive technologies for users with disabilities.

To access the following Sun resources, go to <http://www.sun.com>:

- Downloads of Sun products
- Services and solutions
- Support (including patches and updates)
- Training
- Research
- Communities (for example, Sun Developer Network)

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to <http://docs.sun.com> and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 819-3658.

Getting Started

This chapter describes the Sun Java™ System Application Server system administration. Administering the Application Server includes many tasks such as creating and deploying applications, monitoring and managing performance, and diagnosing and troubleshooting problems. This chapter contains following sections:

- “About the Sun Java System Application Server” on page 29
- “About the Administration Console” on page 35
- “Configuring the Application Server” on page 36
- “Configuration Changes” on page 38
- “Further Information” on page 39

About the Sun Java System Application Server

The Sun Java System Application Server Platform Edition is a Java EE platform compatible server for the development and deployment of Java EE applications and Java Web Services. Production use of this server is free of charge. This section contains the following topics:

- “What is the Application Server?” on page 29
- “Application Server Architecture” on page 31
- “Tools for Administration” on page 34

What is the Application Server?

The Sun Java System Application Server is a fully-featured, Java EE platform application server providing the foundation for building reliable, scalable, and manageable applications. With its comprehensive set of features and support for component-based development, the Sun Java System Application Server provides the underlying core functionality necessary for the development and deployment of business-driven applications.

- **Improved Administration** — Application Server provides remote secure management using a browser-based Admin Console and a scriptable command-line interface. The new Admin Console features include:
 - Comprehensive web services management and development support
 - Enhanced monitoring, visualization and diagnostic tools
 - Enhancements to the Java Management Extensions (JMX™) MBean API to provide for remote, secure, programmatic administration and monitoring using JMX
- **Improved Performance** — Application deployment is faster, which makes iterative application development quicker and easier. The runtime performance of EJB components is also improved.
- **Expanded Platform Support** — The Application Server supports additional operating systems, databases, locales, and hardware (see the Release Notes for the latest list of supported platforms).
- **Migration and Upgrade Tools** — These tools enable you to verify Java EE applications for standards conformance and portability, help with migrations from other Java EE application servers, and aid in upgrading from previous versions of the Application Server.
- **Java 2 Standard Edition Support** — The Application Server supports Java 2 Standard Edition, which includes enhanced management and monitoring features and many performance and scalability improvements.
- **JDBC Drivers** — The Application Server includes DataDirect™ JDBC™ drivers for major databases. You can use these drivers for deployment.
- **Web Services Security** — Container message security mechanisms implement message-level authentication (for example, XML digital signature and encryption) of SOAP web services invocations using the X.509 and username/password profiles of the OASIS WS-Security standard.
- **WS-I Profile 1.1** — As mandated by the Java EE specification, Web Services Interoperability (WS-I) Basic Profile 1.1 enables interoperability for web services applications.
- **Backend Connectivity with iWay Adapters** — Sun Microsystems supports twenty-two iWay adapters to key backend systems to help customers leverage existing applications from the Application Server environment. These adapters support the Java EE Connector Architecture 1.5 specification and web services standards (SOAP), and include developer tools to reduce time to connect to backend applications.

Key Features

The Sun Java System Application Server includes the following key features:

- **Java EE Platform Support** — This release implements all of the specifications covered by the Java EE platform. For a complete list of the Java EE technologies included, see “[Java EE 5 Platform APIs](#)” in *Sun Java System Application Server Platform Edition 9 Release Notes*.

- **JavaServer Faces 1.2** — Application Server supports JavaServer Faces, which simplifies building user interfaces for Java Server applications. Developers can quickly build web applications by assembling reusable user-interface components in a page, connecting the components to an application data source, and wiring client-generated events to server-side event handlers.
- **JavaServer Pages Standard Tag Library 1.2** — The Application Server supports the JavaServer Pages Standard Tag Library (JSTL) 1.2, which encapsulates core functionality common to many JSP applications.
- **Administrative Tools** — The Application Server includes a command-line tool and a browser-based Administration Console graphical user interface.
- **High Performance Message Delivery** — This product supports concurrent message delivery with the Sun Java System Message Queue software.
- **Developer Tool Integration** — The Application Server supports the NetBeans Integrated Development Environment and Sun Java Studio Creator.

Application Server Architecture

This section describes [Figure 1–1](#), which shows the high-level architecture of the Application Server.

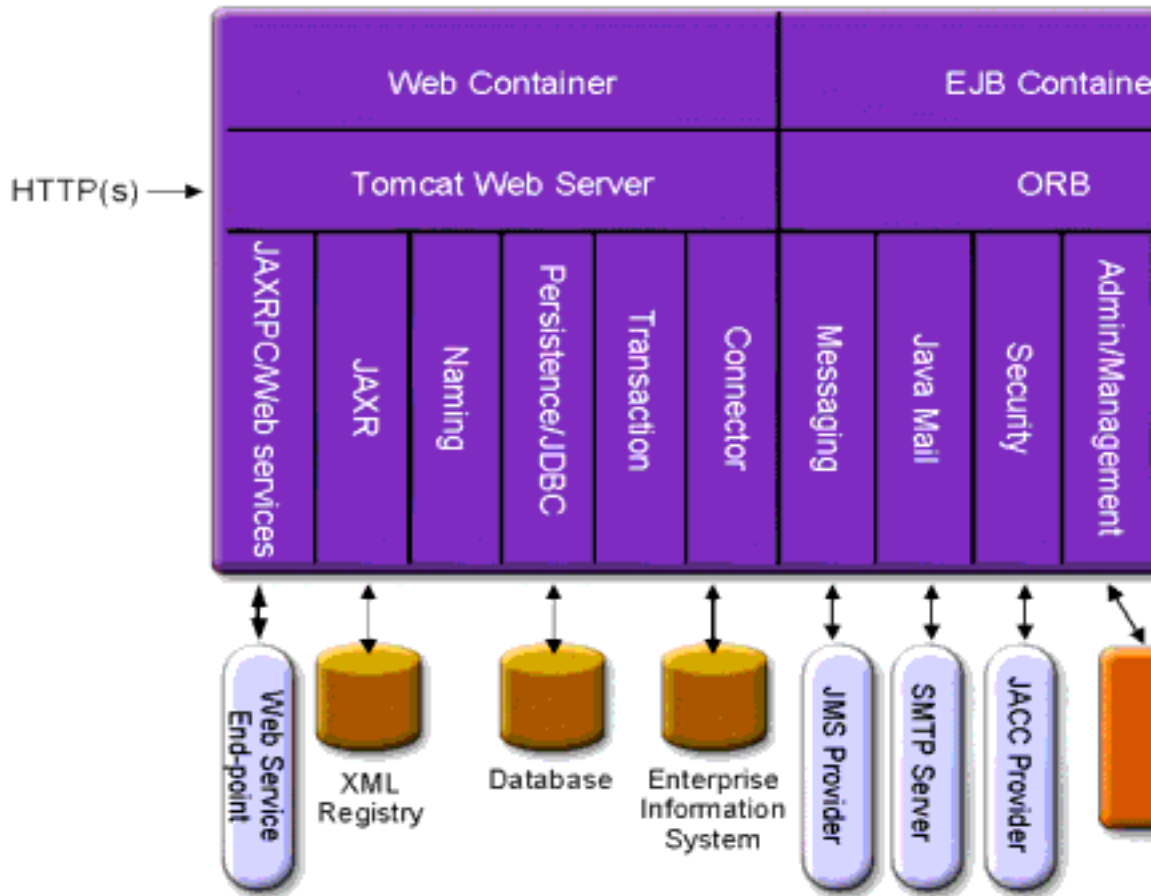


FIGURE 1-1 Application Server Architecture

- Containers** — A container is a runtime environment that provides services such as security and transaction management to Java EE components. Figure 1-1 shows the two types of Java EE containers: Web and EJB. Web components, such as JSP pages and servlets, run within the Web container. Enterprise beans, the components of EJB technology, run within the EJB container.
- Client Access** — At runtime, browser clients access Web applications by communicating with the Web server via HTTP, the protocol used throughout the internet. The HTTPS protocol is for applications that require secure communication. Enterprise bean clients communicate with the Object Request Broker (ORB) through the IIOP or IIOP/SSL (secure) protocols. The Application Server has separate listeners for the HTTP, HTTPS, IIOP, and IIOP/SSL protocols. Each listener has exclusive use of a specific port number.

- **Web Services** — On the Java EE platform, it is possible to deploy a Web application that provides a Web service implemented by Java API for XML-Based RPC (JAX-RPC). A Java EE application or component can also be a client to other Web services. Applications access XML registries through the Java API for XML Registries (JAXR).
- **Services for Applications** — The Java EE platform was designed so that the containers provide services for applications. [Figure 1-1](#) shows the following services:
 - **Naming** — A naming and directory service binds objects to names. A Java EE application locates an object by looking up its JNDI name. JNDI stands for the Java Naming and Directory Interface API.
 - **Security** — The Java Authorization Contract for Containers (JACC) is a set of security contracts defined for the Java EE containers. Based on the client's identity, the containers restrict access to the container's resources and services.
- **Transaction management** — A transaction is an indivisible unit of work. For example, transferring funds between bank accounts is a transaction. A transaction management service ensures that a transaction either completes fully or is rolled back.

Access to External Systems

The Java EE platform enables applications to access systems that are outside of the application server. Applications connect to these systems through objects called resources. One of the responsibilities of an administrator is resource configuration. The Java EE platform enables access to external systems through the following APIs and components:

- **JDBC** — A database management system (DBMS) provides facilities for storing, organizing, and retrieving data. Most business applications store data in relational databases, which applications access via the JDBC API. The information in databases is often described as persistent because it is saved on disk and exists after the application ends. The Application Server bundle includes the PointBase DBMS.
- **Messaging** — Messaging is a method of communication between software components or applications. A messaging client sends messages to, and receives messages from, any other client. Applications access the messaging provider through the Java Messaging Service (JMS) API. The Application Server includes a JMS provider.
- **Connector** — The Java EE Connector architecture enables integration between Java EE applications and existing Enterprise Information Systems (EIS). An application accesses an EIS through a portable Java EE component called a connector or resource adapter.
- **JavaMail** — Through the JavaMail API, applications connect to an SMTP server in order to send and receive email.
- **Server Administration** — The lower right-hand corner of [Figure 1-1](#) shows some of the tasks performed by the administrator of the Application Server. For example, an administrator deploys (installs) applications and monitors the server's performance. These tasks are performed with the administration tools provided by the Application Server.

Tools for Administration

The Application Server includes three administrative tools:

- “Admin Console” on page 34
- “asadmin Utility” on page 34
- “Application Server Management Extension (AMX)” on page 35

Admin Console

The Admin Console is a browser-based tool that features an easy-to-navigate interface and online help. The administration server must be running to use the Admin Console.

When the Application Server was installed, you chose a port number for the server, or used the default port of 4848. You also specified a user name and master password.

To start the Admin Console, in a web browser type:

```
http://hostname:port
```

For example:

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

If the Admin Console is running on the machine on which the Application Server was installed, specify `localhost` for the host name.

On Windows, start the Application Server Admin Console from the Start menu.

The installation program creates the default administrative domain (named `domain1`) with the default port number 4848, as well as an instance separate from the domain administration server (DAS). After installation, additional administration domains can be created. Each domain has its own domain administration server, which has a unique port number. When specifying the URL for the Admin Console, be sure to use the port number for the domain to be administered.

asadmin Utility

The `asadmin` utility is a command-line tool. Use the `asadmin` utility and the commands associated with it to perform the same set of tasks that can be performed in the Admin Console. For example, start and stop domains, configure the server, and deploy applications.

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

To start the `asadmin` utility:

```
$ asadmin
```

To list the commands available within `asadmin`:

```
asadmin> help
```

It is also possible to issue an `asadmin` command at the shell's command prompt:

```
$ asadmin help
```

To view a command's syntax and examples, type `help` followed by the command name. For example:

```
asadmin> help create-jdbc-resource
```

The `asadmin help` information for a given command displays the UNIX man page of the command. These man pages are also available in HTML format.

Application Server Management Extension (AMX)

The Application Server Management eXtension is an API that exposes all of the Application Server configuration and monitoring JMX managed beans as easy-to-use client-side dynamic proxies implementing the AMX interfaces.

For more information on using the Application Server Management Extension, see [Chapter 20, "Using the Application Server Management Extensions," in *Sun Java System Application Server Platform Edition 9 Developer's Guide*](#).

About the Administration Console

The Application Server Admin Console is a Web browser-based, graphical user interface that you use to manage the application server domain. An Application Server domain is a logically related group of Application Server resources that you manage as a unit. A domain includes one or more Application Servers. You deploy and manage your applications as part of a domain.

One instance of the Application Server in the domain is configured as a Domain Administration Server (DAS). The Domain Administration Server provides a central point for managing an Application Server domain. In a domain with only a single Application Server instance, that server functions as both a Domain Administration Server and the management server. The Domain Administration Server hosts the Administration Console, which is a Web application accessible from any supported Web browser with network access to the Administration Server.

Use the Administration Console to:

- Configure, start, and stop Application Server instances
- Configure Application Server services, such as database connectivity (JDBC) and messaging (JMS)

- Configure security parameters, including managing users, groups, and roles
- Configure and deploy your applications
- Monitor server and application performance
- View server and domain log files

Using Online Help

The Admin Console's online help is context-sensitive: When clicking the Help link in the upper right corner, the help browser window displays a topic related to the current Admin Console page.

The online help includes conceptual topics that are not context-sensitive. To view one of these topics, select it from the table of contents in the help browser window.

Configuring the Application Server

Application Server domains are logical or physical units created to help the administrator manage a system configuration. A domain is broken down into smaller units including instances. A server instance is a single Java Virtual Machine (JVM) that runs the Application Server on a single physical machine. Each domain has one or more instance.

Creating a Domain

Domains are created using the `create-domain` command. The following example command creates a domain named `mydomain`. The administration server listens on port 1234 and the administrative user name is `admin`. The command prompts for the administrative and master passwords.

```
$ asadmin create-domain --adminport 80 --adminuser admin mydomain
```

To start the Admin Console for `mydomain` domain, in a browser, enter the following URL:

```
http://hostname:80
```

For the preceding `create-domain` example, the domain's log files, configuration files, and deployed applications now reside in the following directory:

```
domain-root-dir/mydomain
```

To create the domain's directory in another location, specify the `--domaindir` option. For the full syntax of the command, type `asadmin help create-domain`.

Deleting a Domain

Domains are deleted using the `asadmin delete-domain` command. Only the operating system user (or root) who can administer the domain can execute this command successfully. To delete a domain named `mydomain`, for example, type the following command:

```
$ asadmin delete-domain mydomain
```

Listing Domains

The domains created on a machine can be found using the `asadmin list-domains` command. To list the domains in the default *domain-root-dir* directory, type this command:

```
$ asadmin list-domains
```

To list domains that were created in other directories, specify the `--domain-dir` option.

Starting the Domain

When starting a domain, the administration server and application server instance are started. Once the application server instance is started it runs constantly, listening for and accepting requests.

To start a domain, type the `asadmin start-domain` command and specify the domain name. For example, to start the default domain (`domain1`), type the following:

```
$ asadmin start-domain --user admin domain1
```

For the full command syntax, type `asadmin help start-domain`. If the password data is omitted, you are prompted to supply it.

On Windows, to start the default domain:

From the Windows Start Menu, select Programs -> Sun Microsystems -> Application Server -> Start Admin Server.

Stopping the Domain

Stopping a domain shuts down its administration server and application server instance. When stopping a domain, the server instance stops accepting new connections and then waits for all outstanding connections to complete. This process takes a few seconds because the server instance must complete its shutdown process. While the domain is stopped, the Admin Console or most `asadmin` commands cannot be used.

To stop a domain, type the `asadmin stop-domain` command and specify the domain name. For example, to stop the default domain (`domain1`), type the following:

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

If there is only one domain, then the domain name is optional. For the full syntax, type `asadmin help stop-domain`.

On Windows, to stop the default domain:

From the Start menu select Programs -> Sun Microsystems -> Application Server-> Stop Admin Server.

Configuration Changes

When making configuration changes, you may need to restart the server for the changes to take effect. The following sections identify when you will need to restart the server for each configuration change:

- [“Changing Application Server Configuration” on page 38](#)
- [“Ports in the Application Server” on page 38](#)

Changing Application Server Configuration

When making any of these configuration changes, restart the server for the changes to take effect:

- Changing JVM options
- Changing port numbers
- Managing HTTP, IIOP, and JMS services
- Managing thread pools

With dynamic configuration, most changes take effect while the server is running. To make the following configuration changes, do *NOT* restart the server:

- Deploying and undeploying applications
- Adding or removing JDBC, JMS, and Connector resources and pools
- Changing logging levels
- Adding file realm users
- Changing monitoring levels
- Enabling and disabling resources and applications

Ports in the Application Server

The following table describes the port listeners of the Application Server.

TABLE 1-1 Application Server Listeners that Use Ports

Listener	Default Port Number	Description
Administrative server	4848	A domain's administrative server is accessed by the Admin Console and the <code>asadmin</code> utility. For the Admin Console, specify the port number in the URL of the browser. When executing an <code>asadmin</code> command remotely, specify the port number with the <code>--port</code> option.
HTTP	8080	The Web server listens for HTTP requests on a port. To access deployed Web applications and services, clients connect to this port.
HTTPS	8181	Web applications configured for secure communications listen on a separate port.
IIOP		Remote clients of enterprise beans (EJB components) access the beans through the IIOP listener.
IIOP, SSL		Another port is used by the IIOP listener configured for secure communications.
IIOP, SSL and mutual authentication		Another port is used by the IIOP listener configured for mutual (client and server) authentication.

Further Information

- [Sun Microsystems Worldwide Training](#) - Over 250,000 students each year are trained by Sun and its authorized centers through Web-based courses and at over 250 training sites located in more than 60 countries.
- [The Java EE 5 Tutorial](#) — written for developers, the tutorial has administrative instructions for configuring JMS, setting up JavaMail resources, and managing security.
- [Sun Java System Application Server Platform Edition 9 Developer's Guide](#) contains development information that is specific to the Application Server.
- [Sun Java System Application Server Platform Edition 9 Reference Manual](#) provides a reference all the application server command-line utilities including the `asadmin` commands.
- [Sun Java System Application Server Platform Edition 9 Release Notes](#) provide the most up-to-date information on the product release.
- Sun Product Documentation - To access all of our product documentation, see <http://docs.sun.com>.
- [Java EE Documentation page](#)- Contains links to the technical documentation for the Java EE platform: .
- [Sun Java System Application Server Platform Edition 9 Quick Start Guide](#) shows you how to deploy and run a simple Web application. In addition to being online, the guide is provided in the `install-dir/docs/QuickStart.html` file.

JDBC Resources

A JDBC resource (data source) provides applications with a means of connecting to a database. Typically, the administrator creates a JDBC resource for each database accessed by the applications deployed in a domain. (However, more than one JDBC resource can be created for a database.)

This chapter contains the following sections:

- [“About JDBC Resources” on page 41](#)
- [“About JDBC Connection Pools” on page 42](#)
- [“How JDBC Resources and Connection Pools Work Together” on page 43](#)
- [“Configurations for Specific JDBC Drivers” on page 43](#)

About JDBC Resources

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API.

Creating a JDBC Resource

To create a JDBC resource, specify a unique JNDI name that identifies the resource. Expect to find the JNDI name of a JDBC resource in `java:comp/env/jdbc` subcontext. For example, the JNDI name for the resource of a payroll database could be `java:comp/env/jdbc/payrolldb`. Because all resource JNDI names are in the `java:comp/env` subcontext, when specifying the JNDI name of a JDBC resource in the Admin Console, enter only `jdbc/name`. For example, for a payroll database specify `jdbc/payrolldb`.

To create a JDBC resource using the Admin Console, select **Resources > JDBC Resources**. Specify the resources settings as follows:

- **JNDI Name:** Specify a unique name. The JNDI name organizes and locates components within a distributed computing environment similarly to the way that card catalogs organize and represent locations of books in a library. Consequently, the JNDI name becomes an important method of accessing the JDBC resource. By convention, the name begins with the `jdbc/` string. For example: `jdbc/payrolldb`. Don't forget the forward slash.
- **Pool Name:** Choose the connection pool to be associated with the new JDBC resource.
- **Description:** Type a short description of the resource.
- **Status:** If you want the resource to be unavailable, deselect the Enabled checkbox. By default, the resource is available (enabled) as soon as it is created.

About JDBC Connection Pools

To create a JDBC resource, specify the connection pool with which it is associated. Multiple JDBC resources can specify a single connection pool.

A JDBC connection pool is a group of reusable connections for a particular database. Because creating each new physical connection is time consuming, the server maintains a pool of available connections to increase performance. When an application requests a connection, it obtains one from the pool. When an application closes a connection, the connection is returned to the pool.

Creating a JDBC Connection Pool

When creating a connection pool, you are actually defining the aspects of a connection to a specific database. Before creating the pool, you must first install and integrate the JDBC driver. The properties of connection pools can vary with different database vendors. Some common properties are the database's name (URL), user name, and password.

Certain data specific to the JDBC driver and the database vendor must be entered. Before proceeding, gather the following information:

- Database vendor name
- Resource type, such as `javax.sql.DataSource` (local transactions only)
`javax.sql.XADataSource` (global transactions)
- Data source class name: If the JDBC driver has a Datasource class for the resource type and database, then the value of the Datasource Classname field is required.
- Required properties, such as the database name (URL), user name, and password

How JDBC Resources and Connection Pools Work Together

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.

At runtime, here's what happens when an application connects to a database:

1. The application gets the JDBC resource (data source) associated with the database by making a call through the JNDI API.
Given the resource's JNDI name, the naming and directory service locates the JDBC resource. Each JDBC resource specifies a connection pool.
2. Via the JDBC resource, the application gets a database connection.
Behind the scenes, the application server retrieves a physical connection from the connection pool that corresponds to the database. The pool defines connection attributes such as the database name (URL), user name, and password.
3. Now that it's connected to the database, the application can read, modify, and add data to the database.
The applications access the database by making calls to the JDBC API. The JDBC driver translates the application's JDBC calls into the protocol of the database server.
4. When it's finished accessing the database, the application closes the connection.
The application server returns the connection to the connection pool. Once it's back in the pool, the connection is available for the next application.

Configurations for Specific JDBC Drivers

Application Server is designed to support connectivity to any database management system with a corresponding JDBC driver. The following JDBC driver and database combinations are supported. These combinations have been tested with Application Server and are found to be Java EE compatible. They are also supported for CMP.

- “Java DB Type 4 Driver” on page 44
- “Sun Java System JDBC Driver for DB2 Databases” on page 45
- “Sun Java System JDBC Driver for Oracle 9i and 10g Databases” on page 45
- “Sun Java System JDBC Driver for Microsoft SQL Server Databases” on page 46
- “Sun Java System JDBC Driver for Sybase Databases” on page 46
- “IBM DB2 8.1 and 8.2 Type 2 Driver” on page 47
- “Oracle Thin Type 4 Driver for Oracle 9i and 10g Databases” on page 47
- “Microsoft SQL Server JDBC Driver” on page 48
- “PostgreSQL JDBC Driver” on page 49
- “MM MySQL Type 4 Driver (Non-XA)” on page 49

For an up to date list of currently supported JDBC drivers, see the [Sun Java System Application Server Platform Edition 9 Release Notes](#).

Other JDBC drivers can be used with Application Server , but Java EE compliance tests have not been completed with these drivers. Although Sun offers no product support for these drivers, Sun offers limited support of the use of these drivers with Application Server .

- “MM MySQL Type 4 Driver (XA Only)” on page 50
- “JConnect Type 4 Driver for Sybase ASE 12.5 and 15 Databases” on page 51
- “Inet Oraxo JDBC Driver for Oracle 9i and 10g Databases” on page 51
- “Inet Merlia JDBC Driver for Microsoft SQL Server Databases” on page 52
- “Inet Sybelux JDBC Driver for Sybase Databases” on page 53
- “OCI Oracle Type 2 Driver for Oracle 9i and 10g Databases” on page 53
- “IBM Informix Type 4 Driver” on page 54

Note – An Oracle database user running the capture - schema command needs ANALYZE ANY TABLE privileges if that user does not own the schema. These privileges are granted to the user by the database administrator. For information about capture - schema, see [Sun Java System Application Server Platform Edition 9 Reference Manual](#).

Java DB Type 4 Driver

The Java DB database is based on the [Derby database from Apache](#) (<http://db.apache.org/derby/manuals>). The Java DB JDBC driver is included with the Application Server by default.

The JAR file for the Java DB driver is `derbyclient.jar`.

Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** JavaDB
- **DataSource Classname:** Specify one of the following:

```
org.apache.derby.jdbc.ClientDataSource  
org.apache.derby.jdbc.ClientXADataSource
```

- **Properties:**

- **user** - Specify the database user.

This is only necessary if Java DB is configured to use authentication. Java DB does *not* use authentication by default. When the user is provided, it is the name of the schema where the tables reside.

- **password** - Specify the database password.
This is only necessary if Java DB is configured to use authentication.
- **databaseName** - Specify the name of the database.
- **serverName** - Specify the host name or IP address of the database server.
- **portNumber** - Specify the port number of the database server if it is different from the default.
- **URL:** `jdbc:derby://serverName:portNumber/databaseName;create=true`
Include the `;create=true` part only if you want the database to be created if it does not exist.

Sun Java System JDBC Driver for DB2 Databases

The JAR files for this driver are `smbase.jar`, `smdb2.jar`, and `smutil.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** DB2
- **DataSource Classname:** `com.sun.sql.jdbcx.db2.DB2DataSource`
- **Properties:**
 - **serverName** - Specify the host name or IP address of the database server.
 - **portNumber** - Specify the port number of the database server.
 - **databaseName** - Set as appropriate.
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
- **URL:** `jdbc:sun:db2://serverName:portNumber;databaseName=databaseName`

Sun Java System JDBC Driver for Oracle 9i and 10g Databases

The JAR files for this driver are `smbase.jar`, `smoracle.jar`, and `smutil.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** Oracle
- **DataSource Classname:** `com.sun.sql.jdbcx.oracle.OracleDataSource`
- **Properties:**

- **serverName** - Specify the host name or IP address of the database server.
- **portNumber** - Specify the port number of the database server.
- **SID** - Set as appropriate.
- **user** - Set as appropriate.
- **password** - Set as appropriate.
- **URL:** `jdbc:sun:oracle://serverName[:portNumber][;SID=databaseName]`

Sun Java System JDBC Driver for Microsoft SQL Server Databases

The JAR files for this driver are `smbase.jar`, `smsqlserver.jar`, and `smutil.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `mssql`
- **DataSource Classname:** `com.sun.sql.jdbcx.sqlserver.SQLServerDataSource`
- **Properties:**
 - **serverName** - Specify the host name or IP address and the port of the database server.
 - **portNumber** - Specify the port number of the database server.
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
 - **selectMethod** - Set to `cursor`.
- **URL:** `jdbc:sun:sqlserver://serverName[:portNumber]`

Sun Java System JDBC Driver for Sybase Databases

The JAR files for this driver are `smbase.jar`, `smsybase.jar`, and `smutil.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Sybase`
- **DataSource Classname:** `com.sun.sql.jdbcx.sybase.SybaseDataSource`
- **Properties:**
 - **serverName** - Specify the host name or IP address of the database server.
 - **portNumber** - Specify the port number of the database server.

- **databaseName** - Set as appropriate. This is optional.
- **user** - Set as appropriate.
- **password** - Set as appropriate.
- **URL**: `jdbc:sun:sybase://serverName[:portNumber]`

IBM DB2 8.1 and 8.2 Type 2 Driver

The JAR files for the DB2 driver are `db2jcc.jar`, `db2jcc_license_cu.jar`, and `db2java.zip`. Set environment variables as follows:

```
LD_LIBRARY_PATH=/usr/db2user/sql/lib:${j2ee.home}/lib
DB2DIR=/opt/IBM/db2/V8.1
DB2INSTANCE=db2user
INSTHOME=/usr/db2user
VWSPATH=/usr/db2user/sql/lib
THREADS_FLAG=native
```

Configure the connection pool using the following settings:

- **Name**: Use this name when you configure the JDBC resource later.
- **Resource Type**: Specify the appropriate value.
- **Database Vendor**: DB2
- **DataSource Classname**: `com.ibm.db2.jcc.DB2SimpleDataSource`
- **Properties**:
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
 - **databaseName** - Set as appropriate.
 - **driverType** - Set to 2.
 - **deferPrepares** - Set to false.

Oracle Thin Type 4 Driver for Oracle 9i and 10g Databases

The JAR file for the Oracle driver is `ojdbc14.jar`. Configure the connection pool using the following settings:

- **Name**: Use this name when you configure the JDBC resource later.
- **Resource Type**: Specify the appropriate value.
- **Database Vendor**: Oracle
- **DataSource Classname**: Specify one of the following:

```
oracle.jdbc.pool.OracleDataSource  
oracle.jdbc.xa.client.OracleXADataSource
```

- **Properties:**

- **user** - Set as appropriate.
- **password** - Set as appropriate.
- **URL** - Specify the complete database URL using the following syntax:

```
jdbc:oracle:thin:[user/password]@host[:port]/service
```

For example:

```
jdbc:oracle:thin:@localhost:1521:customer_db
```

- **xa-driver-does-not-support-non-tx-operations** - Set to the value `true`. Optional: only needed if both non-XA and XA connections are retrieved from the same connection pool. Might degrade performance.

As an alternative to setting this property, you can create two connection pools, one for non-XA connections and one for XA connections.

Note – You must set the `oracle-xa-recovery-workaround` property in the Transaction Service for recovery of global transactions to work correctly. For details, see [“Workarounds for Specific Databases” on page 156](#).

When using this driver, it is not possible to insert more than 2000 bytes of data into a column. To circumvent this problem, use the OCI driver (JDBC type 2).

Microsoft SQL Server JDBC Driver

The JAR file for this driver is `sqljdbc.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `mssql`
- **DataSource Classname:** Specify one of the following:

```
com.microsoft.sqlserver.jdbc.SQLServerDataSource  
com.microsoft.sqlserver.jdbc.SQLServerXADataSource
```

- **Properties:**

- **serverName** - Specify the host name or IP address and the port of the database server. This is optional. This could be a DNS or IP address, or it could be `localhost` or `127.0.0.1` for the local computer.

- **instanceName** - Specify the instance to connect to on serverName. This is optional. If not specified, a connection to the default instance is made.
- **portNumber** - Specify the port number of the database server. This is optional. The default is 1433. For optimal connection performance, set the portNumber when connecting to a named instance. This avoids a round trip to the server to determine the port number.
- **user** - Set as appropriate.
- **password** - Set as appropriate.
- **selectMethod** - Set to cursor.
- **Additional properties** - For additional optional properties you can set, see [Setting the Connection Properties](http://msdn2.microsoft.com/en-us/library/ms378988%28SQL.90%29.aspx) (<http://msdn2.microsoft.com/en-us/library/ms378988%28SQL.90%29.aspx>).
- **URL:**
jdbc:sqlserver://[serverName][\instanceName][:portNumber][:property=value]

PostgreSQL JDBC Driver

The JAR file for the PostgreSQL driver is `postgresql-version.jdbc3.jar`, for example, `postgresql-8.2dev-500.jdbc3.jar`. For more information, see <http://jdbc.postgresql.org>. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** PostgreSQL
- **DataSource Classname:** `org.postgresql.ds.PGSimpleDataSource`
- **Properties:**
 - **serverName** - Specify the host name or IP address of the database server.
 - **portNumber** - Specify the port number of the database server.
 - **databaseName** - Set as appropriate.
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.

MM MySQL Type 4 Driver (Non-XA)

The JAR file for the MySQL driver is `mysql-connector-java-version-bin-g.jar`, for example, `mysql-connector-java-3.1.12-bin-g.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.

- **Database Vendor:** `mysql`
- **DataSource Classname:** Specify one of the following:

`com.mysql.jdbc.jdbc2.optional.MysqlDataSource`

- **Properties:**
 - **serverName** - Specify the host name or IP address of the database server.
 - **port** - Specify the port number of the database server.
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
 - **databaseName** - Set as appropriate.
 - **URL** - If you are using global transactions, you can set this property instead of `serverName`, `port`, and `databaseName`.

The MM MySQL Type 4 driver doesn't provide a method to set the required `relaxAutoCommit` property, so you must set it indirectly by setting the **URL** property:

```
jdbc:mysql://host:port/database?relaxAutoCommit="true"
```

MM MySQL Type 4 Driver (XA Only)

The JAR file for the MySQL driver is `mysql-connector-java-version-bin-g.jar`, for example, `mysql-connector-java-3.1.12-bin-g.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `mysql`
- **DataSource Classname:** Specify one of the following:

`com.mysql.jdbc.jdbc2.optional.MysqlXADataSource`

- **Properties:**
 - **serverName** - Specify the host name or IP address of the database server.
 - **port** - Specify the port number of the database server.
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
 - **databaseName** - Set as appropriate.
 - **URL** - If you are using global transactions, you can set this property instead of `serverName`, `port`, and `databaseName`.

The MM MySQL Type 4 driver doesn't provide a method to set the required `relaxAutoCommit` property, so you must set it indirectly by setting the **URL** property:

```
jdbc:mysql://host:port/database?relaxAutoCommit="true"
```

JConnect Type 4 Driver for Sybase ASE 12.5 and 15 Databases

The JAR file for the Sybase driver is `jconn2.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** Sybase
- **DataSource Classname:** Specify one of the following:

```
com.sybase.jdbc2.jdbc.SybDataSource
com.sybase.jdbc2.jdbc.SybXADatasource
```

- **Properties:**
 - **serverName** - Specify the host name or IP address of the database server.
 - **portNumber** - Specify the port number of the database server.
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
 - **databaseName** - Set as appropriate. Do not specify the complete URL, only the database name.
 - **BE_AS_JDBC_COMPLIANT_AS_POSSIBLE** - Set to `true`.
 - **FAKE_METADATA** - Set to `true`.

Inet Oraxo JDBC Driver for Oracle 9i and 10g Databases

The JAR file for the Inet Oracle driver is `Oranxo.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** Oracle
- **DataSource Classname:** `com.inet.ora.OraDataSource`

- **Properties:**

- **user** - Specify the database user.
- **password** - Specify the database password.
- **serviceName** - Specify the URL of the database. The syntax is as follows:

```
jdbc:inetora:server:port:dbname
```

For example:

```
jdbc:inetora:localhost:1521:payrolldb
```

In this example, `localhost` is the host name of the machine running the Oracle server, `1521` is the Oracle server's port number, and `payrolldb` is the SID of the database. For more information about the syntax of the database URL, see the Oracle documentation.

- **serverName** - Specify the host name or IP address of the database server.
- **port** - Specify the port number of the database server.
- **streamstolob** - If the size of BLOB or CLOB data types exceeds 4 KB and this driver is used for CMP, this property must be set to `true`.
- **xa-driver-does-not-support-non-tx-operations** - Set to the value `true`. Optional: only needed if both non-XA and XA connections are retrieved from the same connection pool. Might degrade performance.

As an alternative to setting this property, you can create two connection pools, one for non-XA connections and one for XA connections.

Inet Merlia JDBC Driver for Microsoft SQL Server Databases

The JAR file for the Inet Microsoft SQL Server driver is `Merlia.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `mssql`
- **DataSource Classname:** `com.inet.tds.TdsDataSource`
- **Properties:**
 - **serverName** - Specify the host name or IP address and the port of the database server.
 - **port** - Specify the port number of the database server.
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.

Inet Sybelux JDBC Driver for Sybase Databases

The JAR file for the Inet Sybase driver is `Sybelux.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** Sybase
- **DataSource Classname:** `com.inet.syb.SybDataSource`
- **Properties:**
 - **serverName** - Specify the host name or IP address of the database server.
 - **portNumber** - Specify the port number of the database server.
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
 - **databaseName** - Set as appropriate. Do not specify the complete URL, only the database name.

OCI Oracle Type 2 Driver for Oracle 9i and 10g Databases

The JAR file for the OCI Oracle driver is `ojdbc14.jar`. Make sure that the shared library is available through `LD_LIBRARY_PATH` and that the `ORACLE_HOME` property is set. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** Oracle
- **DataSource Classname:** Specify one of the following:
 - `oracle.jdbc.pool.OracleDataSource`
 - `oracle.jdbc.xa.client.OracleXADataSource`
- **Properties:**
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
 - **URL** - Specify the complete database URL using the following syntax:

```
jdbc:oracle:oci:[user/password]@host[:port]/service
```

For example:

```
jdbc:oracle:oci:@localhost:1521:customer_db
```

- **xa-driver-does-not-support-non-tx-operations** - Set to the value `true`. Optional: only needed if both non-XA and XA connections are retrieved from the same connection pool. Might degrade performance.

As an alternative to setting this property, you can create two connection pools, one for non-XA connections and one for XA connections.

IBM Informix Type 4 Driver

Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** Informix
- **DataSource Classname:** Specify one of the following:

```
com.informix.jdbcx.IfxDatasource  
com.informix.jdbcx.IfxxDataSource
```

- **Properties:**
 - **serverName** - Specify the Informix database server name.
 - **portNumber** - Specify the port number of the database server.
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
 - **databaseName** - Set as appropriate. This is optional.
 - **IfxIFXHost** - Specify the host name or IP address of the database server.

Configuring Java Message Service Resources

This chapter describes how to configure resources for applications that use the Java Message Service (JMS) API. It contains the following sections:

- “About JMS Resources” on page 55
- “JMS Connection Factories” on page 57
- “JMS Destination Resources” on page 58
- “JMS Physical Destinations” on page 58
- “JMS Providers” on page 59
- “Foreign JMS Providers” on page 60

About JMS Resources

- “The JMS Provider in the Application Server” on page 55
- “JMS Resources” on page 55
- “The Relationship Between JMS Resources and Connector Resources” on page 57

The JMS Provider in the Application Server

The Application Server implements the Java Message Service (JMS) API by integrating the Sun Java System Message Queue (formerly Sun ONE Message Queue) software into Application Server. For basic JMS API administration tasks, use the Application Server Admin Console. For advanced tasks, use the tools provided in the *MQ-install-dir/imq/bin* directory.

For details about administering Message Queue, see the *Message Queue Administration Guide*.

JMS Resources

The Java Message Service (JMS) API uses two kinds of administered objects:

- Connection factories, objects that allow an application to create other JMS objects programmatically
- Destinations, which serve as the repositories for messages

These objects are created administratively, and how they are created is specific to each implementation of JMS. In the Application Server, you can perform the following tasks:

- Create a connection factory by creating a connection factory resource
- Create a destination by creating two objects:
 - A physical destination
 - A destination resource that refers to the physical destination

JMS applications use the JNDI API to access the connection factory and destination resources. A JMS application normally uses at least one connection factory and at least one destination. To learn what resources to create, study the application or consult with the application developer.

There are three types of connection factories:

- `QueueConnectionFactory` objects, used for point-to-point communication
- `TopicConnectionFactory` objects, used for publish-subscribe communication
- `ConnectionFactory` objects, which can be used for both point-to-point and publish-subscribe communications; these are recommended for new applications

There are two kinds of destinations:

- Queue objects, used for point-to-point communication
- Topic objects, used for publish-subscribe communication

The chapters on JMS in the *J2EE 1.4 Tutorial* provide details on these two types of communication and other aspects of JMS (see <http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html>).

The order in which the resources are created does not matter.

For a Java EE application, specify connection factory and destination resources in the Application Server deployment descriptors as follows:

- Specify a connection factory JNDI name in a `resource-ref` or an `mdb-connection-factory` element.
- Specify a destination resource JNDI name in the `ejb` element for a message-driven bean and in the `message-destination` element.
- Specify a physical destination name in a `message-destination-link` element, within either a `message-driven` element of an enterprise bean deployment descriptor or a `message-destination-ref` element. In addition, specify it in the `message-destination` element. (The `message-destination-ref` element replaces the `resource-env-ref`

element, which is deprecated in new applications.) In the message-destination element of an Application Server deployment descriptor, link the physical destination name with the destination resource name.

See also:

- [“The Relationship Between JMS Resources and Connector Resources” on page 57](#)

The Relationship Between JMS Resources and Connector Resources

The Application Server implements JMS by using a system resource adapter named `jmsra`. When a user creates JMS resources, the Application Server automatically creates connector resources that appear under the Connectors node in the Admin Console’s tree view.

For each JMS connection factory that a user creates, the Application Server creates a connector connection pool and connector resource. For each JMS destination that a user creates, the Application Server creates an admin object resource. When the user deletes the JMS resources, the Application Server automatically deletes the connector resources.

It is possible to create connector resources for the JMS system resource adapter by using the Connectors node of the Admin Console instead of the JMS Resources node.

See also:

- [“JMS Resources” on page 55](#)

JMS Connection Factories

JMS connection factories are objects that allow an application to create other JMS objects programmatically. These administered objects implement the `ConnectionFactory`, `QueueConnectionFactory`, and `TopicConnectionFactory` interfaces. Using the Application Server Admin Console, you can create, edit, or delete a JMS Connection Factory. The creation of a new JMS connection factory also creates a connector connection pool for the factory and a connector resource.

Equivalent `asadmin` command

```
create-jms-resource
```

```
list-jms-resources
```

```
delete-jms-resource
```

JMS Destination Resources

JMS destinations serve as the repositories for messages. Using the Admin Console, you can create, modify or delete JMS Destination Resources. To create a new JMS Destination Resource, select Resources>JMS Resources>Destination Resources. In the Destination Resources page, you can specify the following:

- JNDI Name for the resource. It is a recommended practice to use the naming subcontext prefix `jms/` for JMS resources. For example: `jms/Queue`.
- The resource type, which can be `javax.jms.Topic` or `javax.jms.Queue`.
- Additional properties for the destination resource. For more details about all these settings and the additional properties, refer to the Admin Console Online Help.

Equivalent `asadmin` commands

```
create-jms-resource
```

```
delete-jms-resource
```

JMS Physical Destinations

For production purposes, always create physical destinations. During the development and testing phase, however, this step is not required. The first time that an application accesses a destination resource, Message Queue automatically creates the physical destination specified by the Name property of the destination resource. The physical destination is temporary and expires after a period specified by a Message Queue configuration property.

To create a physical destination from the Admin Console, select Configuration>Physical Destinations. In the Create Physical Destinations page, specify a name for the physical destination and choose the type of destination, which can be `topic` or `queue`. For more details about the fields and properties in the Physical Destinations page, refer the Admin Console Online Help.

Equivalent `asadmin` commands

```
create-jmsdest
```

```
flush-jmsdest
```

```
delete-jmsdest
```

JMS Providers

Configuring General Properties for the JMS Provider

Use the JMS Service page in the Admin Console to configure properties to be used by all JMS connections. In the Admin Console, select Configuration>Java Message Service. In the JMS Service page, you can control the following general JMS settings.

- Select Startup Timeout interval, which indicates the time that Application Server waits for the JMS service to start before aborting the startup.
- Select JMS Service type, which decides whether you manage a JMS Service on a local or a remote host.
- Specify Start Arguments to customize the JMS service startup.
- Select Reconnect checkbox to specify whether the JMS service attempts to reconnect to a message server (or the list of addresses in the AddressList) when a connection is lost.
- Specify Reconnect Interval in terms of number of seconds. This applies for attempts on each address in the AddressList and for successive addresses in the list. If it is too short, this time interval does not give a broker time to recover. If it is too long, the reconnect might represent an unacceptable delay.
- Specify the number of reconnect attempts. In the field, type the number of attempts to connect (or reconnect) for each address in the AddressList before the client runtime tries the next address in the list.
- Choose the default JMS host.
- In the Address List Behavior drop-down list, choose whether connection attempts are in the order of addresses in the AddressList (priority) or in a random order (random).
- In the Address List Iterations field, type the number of times the JMS service iterates through the AddressList in an effort to establish (or reestablish) a connection.
- In the MQ Scheme and MQ Service fields, type the Message Queue address scheme name and the Message Queue connection service name if a non-default scheme or service is to be used.

Values of all these properties can be updated at run time too. However, only those connection factories that are created after the properties are updated, will get the updated values. The existing connection factories will continue to have the original property values. Also, for almost all of the values to take effect, the application server needs to be restarted. The only property that can be updated without having to restart the application server is the default JMS host.

For detailed descriptions of all the settings and properties mentioned above, you can see the Admin Console Online Help.

Accessing Remote Servers

Changing the provider and host to a remote system causes all JMS applications to run on the remote server. To use both the local server and one or more remote servers, create a connection factory resource with the `AddressList` property to create connections that access remote servers.

Equivalent `asadmin` commands

```
set
```

```
jms-ping
```

Foreign JMS Providers

Generic Resource Adapter 1.5 for JMS is a Java EE Connector 1.5 resource adapter that can wrap the JMS client library of external JMS providers such as IBM Websphere MQ, Tibco EMS, and Sonic MQ among others, and thus integrate any JMS provider with a Java EE 1.4 application server such as Sun Java System Application Server. The adapter is a .rar archive that can be deployed and configured using a Java EE 1.4 application server's administration tools.

Configuring the Generic Resource Adapter for JMS

Application Server's administration tools can be used to deploy and configure the generic resource adapter for JMS. This section explains how to configure Generic Resource Adapter for JMS with Sun Java System Application Server.

Overall, the Resource Adapter can be configured to indicate whether the JMS provider supports XA or not. It is also possible to indicate what mode of integration is possible with the JMS provider. Two modes of integration are supported by the resource adapter. The first one uses JNDI as the means of integration. In this case, administered objects are set up in the JMS provider's JNDI tree and will be looked up for use by the generic resource adapter. If that mode is not suitable for integration, it is also possible to use the Java reflection of JMS administered object `Javabeans` classes as the mode of integration.

You can use the Sun Java System Application Server's Admin Console or the CLI to configure the resource adapter. This is not different from configuring any other resource adapter.

▼ To Configure the Generic Resource Adapter

Prior to deploying the resource adapter, JMS client libraries should be made available to the application server. For some JMS providers, client libraries may also include native libraries. In such cases, these native libraries should also be made available to the application server JVM(s).

- 1 **Deploy the generic resource adapter the same way you would deploy a connector module.**
 For steps to do this, refer to the Admin Console Online Help. During deployment, make sure that you specify `install-dir/lib/addons/resourceadapters/genericjmsra/genericra.rar` as the location of the generic resource adapter. Also, you must specify the properties explained in the section [“Resource Adapter Properties” on page 61](#).
- 2 **Create a connector connection pool.**
 For steps to do this, refer to the Admin Console Online help. In the New Connector Connection Pool page, from the Resource Adapter combo box, select `genericra`. Also, in the Connection Definition combo box, select `javax.jms.QueueConnectionFactory`. Additionally, specify the properties explained in the section [“ManagedConnectionFactory Properties” on page 66](#).
- 3 **Create a connector resource.**
 For detailed procedure to do this, you can refer the Admin Console Online Help. In the New Connector Resource page, select the pool you created in the previous step.
- 4 **Create an administered object resource.**
 For detailed procedure to do this, you can refer the Admin Console Online Help. In the New Admin Object Resource page, select `genericra` as the Resource Adapter and `javax.jms.Queue` as the Resource Type. Click Next and in the second page, click Add Property. In the Additional Properties table, specify a new property called `DestinationProperties` with the value `Name\\=clientQueue`. For information on more properties, see the section [“Administered Object Resource Properties” on page 66](#).
- 5 **Make the following changes to the security policy in Sun Java System Application Server.**
 - a. **Modify** `sjsas_home/domains/domain1/config/server.policy` **to add**
`java.util.logging.LoggingPermission "control"`
 - b. **Modify** `sjsas_home/lib/appclient/client.policy` **to add permission**
`javax.security.auth.PrivateCredentialPermission`
`"javax.resource.spi.security.PasswordCredential * *\\"", "read";`

Resource Adapter Properties

The following table presents the properties to be used while creating the resource adapter.

Property Name	Valid Values	Default Value	Description
ProviderIntegrationMode	javabeans/jndi	javabeans	Decides the mode of integration between the resource adapter and the JMS client.
ConnectionFactoryClassName	Name of the class available in the appserver classpath, for example, com.sun.messaging.ConnectionFactory	None	Class name of javax.jms.ConnectionFactory implementation of the JMS client. Used if ProviderIntegrationMode is javabeans.
QueueConnectionFactoryClassName	Name of the class available in the appserver classpath, for example, com.sun.messaging.QueueConnectionFactory	None	Class name of javax.jms.QueueConnectionFactory implementation of the JMS client. Used if ProviderIntegrationMode is javabeans.
TopicConnectionFactoryClassName	Name of the class available in the appserver classpath, for example, com.sun.messaging.TopicConnectionFactory	None	Class name of javax.jms.TopicConnectionFactory implementation of the JMS client. Used if ProviderIntegrationMode is specified as javabeans.
XAConnectionFactoryClassName	Name of the class available in appserver classpath, for example, com.sun.messaging.XAConnectionFactory	None	Class name of javax.jms.ConnectionFactory implementation of the JMS client. Used if ProviderIntegrationMode is specified as javabeans.
XAQueueConnectionFactoryClassName	Name of the class available in appserver classpath, for example, com.sun.messaging.XAQueueConnectionFactory	None	Class name of javax.jms.XAQueueConnectionFactory implementation of the JMS client. Used if ProviderIntegrationMode is specified as javabeans.
XATopicConnectionFactoryClassName	Name of the class available in appserver classpath, for example, com.sun.messaging.XATopicConnectionFactory	None	Class name of javax.jms.XATopicConnectionFactory implementation of the JMS client. Used if ProviderIntegrationMode is javabeans.

<i>Property Name</i>	<i>Valid Values</i>	<i>Default Value</i>	<i>Description</i>
TopicClassName	Name of the class available in appserver classpath , for example, <code>com.sun.messaging.Topic</code>	None	Class Name of <code>javax.jms.Topic</code> implementation of the JMS client. Used if <code>ProviderIntegrationMode</code> is <code>javabeen</code> .
QueueClassName	Name of the class available in appserver classpath , for example, <code>com.sun.messaging.Queue</code>	None	Class Name of <code>javax.jms.Queue</code> implementation of the JMS client. Used if <code>ProviderIntegrationMode</code> is specified as a <code>javabeen</code> .
SupportsXA	True/false	FALSE	Specifies whether the JMS client supports XA or not.
ConnectionFactoryProperties	Name value pairs separated by comma	None	Specifies the <code>javabeen</code> property names and values of the <code>ConnectionFactory</code> of the JMS client. Required only if <code>ProviderIntegrationMode</code> is <code>javabeen</code> .
JndiProperties	Name value pairs separated by comma	None	Specifies the JNDI provider properties to be used for connecting to the JMS provider's JNDI. Used only if <code>ProviderIntegrationMode</code> is <code>jndi</code> .
CommonSetterMethodName	Method name	None	Specifies the common setter method name that some JMS vendors use to set the properties on their administered objects. Used only if <code>ProviderIntegrationMode</code> is <code>javabeen</code> . In the case of Sun Java System Message Queue, this property is named <code>setProperty</code> .
UserName	Name of the JMS user	None	User name to connect to the JMS Provider.

<i>Property Name</i>	<i>Valid Values</i>	<i>Default Value</i>	<i>Description</i>
Password	Password for the JMS user	None	Password to connect to the JMS provider.

<i>Property Name</i>	<i>Valid Values</i>	<i>Default Value</i>	<i>Description</i>
RMPolicy	ProviderManaged or OnePerPhysicalConnection	ProviderManaged	<p>The isSameRM method on an XAResource is used by the Transaction Manager to determine if the Resource Manager instance represented by two XAResources are the same. When RMPolicy is set to ProviderManaged (the default value), the JMS provider is responsible for determining the RMPolicy and the XAResource wrappers in the Generic Resource Adapter merely delegate the isSameRM call to the message queue provider's XA resource implementations. This should ideally work for most message queue products.</p> <p>Some XAResource implementations such as IBM MQ Series rely on a resource manager per physical connection and this causes issues when there is inbound and outbound communication to the same queue manager in a single transaction (for example, when an MDB sends a response to a destination). When RMPolicy is set to OnePerPhysicalConnection, the XAResource wrapper implementation's isSameRM in Generic Resource Adapter would check if both the XAResources use the same physical connection, before delegating to the wrapped objects.</p>

ManagedConnectionFactory Properties

ManagedConnectionFactory properties are specified when a connector-connection-pool is created. All the properties specified while creating the resource adapter can be overridden in a ManagedConnectionFactory. Additional properties available only in ManagedConnectionFactory are given below.

Property Name	Valid Value	Default Value	Description
ClientId	A valid client ID	None	ClientID as specified by JMS 1.1 specification.
ConnectionFactoryJndiName	JNDI Name	None	JNDI name of the connection factory bound in the JNDI tree of the JMS provider. The administrator should provide all connection factory properties (except clientID) in the JMS provider itself. This property name will be used only if ProviderIntegratinMode is jndi.
ConnectionValidationEnabled	true/false	FALSE	If set to true, the resource adapter will use an exception listener to catch any connection exception and will send a CONNECTION_ERROR_OCCURED event to application server.

Administered Object Resource Properties

Properties in this section are specified when an administered object resource is created. All the resource adapter properties can be overridden in an administered resource object. Additional properties available only in the administered object resource are given below.

<i>Property Name</i>	<i>Valid Value</i>	<i>Default Value</i>	<i>Description</i>
DestinationJndiName	JNDI Name	None	JNDI name of the destination bound in the JNDI tree of the JMS provider. The Administrator should provide all properties in the JMS provider itself. This property name will be used only if ProviderIntegrationMode is jndi.
DestinationProperties	Name value pairs separated by comma	None	Specifies the javabean property names and values of the destination of the JMS client. Required only if ProviderIntegrationMode is javabean.

Activation Spec Properties

Properties in this section are specified in the Sun-specific deployment descriptor of MDB as activation-config-properties. All the resource adapter properties can be overridden in an Activation Spec. Additional properties available only in ActivationSpec are given below.

<i>Property Name</i>	<i>Valid Value</i>	<i>Default Value</i>	<i>Description</i>
MaxPoolSize	An integer	8	Maximum size of server session pool internally created by the resource adapter for achieving concurrent message delivery. This should be equal to the maximum pool size of MDB objects.
MaxWaitTime	An integer	3	The resource adapter will wait for the time in seconds specified by this property to obtain a server session from its internal pool. If this limit is exceeded, message delivery will fail.

<i>Property Name</i>	<i>Valid Value</i>	<i>Default Value</i>	<i>Description</i>
SubscriptionDurability	Durable or Non-Durable	Non-Durable	SubscriptionDurability as specified by JMS 1.1 specification.
SubscriptionName		None	SubscriptionName as specified by JMS 1.1 specification.
MessageSelector	A valid message selector	None	MessageSelector as specified by JMS 1.1 specification.
ClientID	A valid client ID	None	ClientID as specified by JMS 1.1 specification.
ConnectionFactoryJndiName	A valid Jndi Name	None	JNDI name of connection factory created in JMS provider. This connection factory will be used by resource adapter to create a connection to receive messages. Used only if ProviderIntegrationMode is configured as jndi.
DestinationJndiName	A valid Jndi Name	None	JNDI name of destination created in JMS provider. This destination will be used by resource adapter to create a connection to receive messages from. Used only if ProviderIntegrationMode is configured as jndi.
DestinationType	javax.jms.Queue or javax.jms.Topic	Null	Type of the destination the MDB will listen to.
DestinationProperties	Name-value pairs separated by comma	None	Specifies the javabean property names and values of the destination of the JMS client. Required only if ProviderIntegrationMode is javabean.

<i>Property Name</i>	<i>Valid Value</i>	<i>Default Value</i>	<i>Description</i>
RedeliveryAttempts	integer		Number of times a message will be delivered if a message causes a runtime exception in the MDB.
RedeliveryInterval	time in seconds		Interval between repeated deliveries, if a message causes a runtime exception in the MDB.
SendBadMessagesToDMD	true/false	False	Indicates whether the resource adapter should send the messages to a dead message destination, if the number of delivery attempts is exceeded.
DeadMessageDestinationJndiName	JNDI name.	None	JNDI name of the destination created in the JMS provider. This is the target destination for dead messages. This is used only if <code>ProviderIntegrationMode</code> is <code>jndi</code> .
DeadMessageDestinationClassName	Class name of destination object.	None	Used if <code>ProviderIntegrationMode</code> is <code>javabeen</code> .
DeadMessageDestinationProperties	Property Value Pairs separated by comma	None	Specifies the javabeen property names and values of the destination of the JMS client. This is required only if <code>ProviderIntegrationMode</code> is <code>javabeen</code> .
ReconnectAttempts	integer		Number of times a reconnect will be attempted in case exception listener catches an error on connection.
ReconnectInterval	time in seconds		Interval between reconnects.

Configuring JavaMail Resources

The Application Server includes the JavaMail API. The JavaMail API is a set of abstract APIs that model a mail system. The API provides a platform-independent and protocol-independent framework to build mail and messaging applications. The JavaMail API provides facilities for reading, composing, and sending electronic messages. Service providers implement particular protocols. Using the JavaMail API you can add email capabilities to your applications. JavaMail provides access from Java applications to Internet Message Access Protocol (IMAP)- and Simple Mail Transfer Protocol (SMTP)-capable mail servers on your network or the Internet. It does not provide mail server functionality; you must have access to a mail server to use JavaMail.

To learn more about the JavaMail API, consult the JavaMail web site at <http://java.sun.com/products/javamail/index.html>

This chapter contains the following section:

- “Creating a JavaMail Session” on page 71

Creating a JavaMail Session

To configure JavaMail for use in Application Server, create a Mail Session in the Application Server Admin Console. This allows server-side components and applications to access JavaMail services with JNDI, using the Session properties you assign for them. When creating a Mail Session, you can designate the mail hosts, transport and store protocols, and the default mail user in the Admin Console so that components that use JavaMail do not have to set these properties. Applications that are heavy email users benefit because the Application Server creates a single Session object and makes it available via JNDI to any component that needs it.

To create a JavaMail session using the Admin Console, select Resources > JavaMail Sessions. Specify the JavaMail settings as follows:

- JNDI Name: The unique name for the mail session. Use the naming sub-context prefix `mail/` for JavaMail resources. For example: `mail/MySession`.

- **Mail Host:** The host name of the default mail server. The connect methods of the Store and Transport objects use this value if a protocol-specific host property is not supplied. The name must be resolvable to an actual host name.
- **Default User:** The user name to provide when connecting to a mail server. The connect methods of the Store and Transport objects use this value if a protocol-specific username property is not supplied.
- **Default Return Address:** The email address of the default user, in the form: *username@host.domain*.
- **Description:** Provide a descriptive statement for the component.
- **Session:** Deselect the Enabled checkbox if you do not want to enable the mail session at this time.

Additionally, define the following Advanced settings only if the mail provider has been re-configured to use a non-default store or transport protocol:

- **Store Protocol:** Defines the Store object communication method to be used. By default, the Store Protocol is `imap`.
- **Store Protocol Class:** Provides the Store communication method class that implements the desired Store protocol. By default, the Store Protocol Class is `com.sun.mail.imap.IMAPStore`.
- **Transport Protocol:** Identifies the transport communication method. By default, the Transport Protocol is `smtp`.
- **Transport Protocol Class:** Defines the communication method for the transport class. By default, the Transport Protocol Class is `com.sun.mail.smtp.SMTPTransport`.
- **Debug:** Select this checkbox to enable extra debugging output, including a protocol trace, for this mail session. If the JavaMail log level is set to `FINE` or finer, the debugging output is generated and is included in the system log file. See [“Configuring Log Levels” on page 186](#) for information about setting the log level.
- **Additional Properties:** Create properties required by applications, such as a protocol-specific host or username property. The [JavaMail API](#) documentation lists the available properties.

JNDI Resources

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

JNDI is the acronym for the Java Naming and Directory Interface API. By making calls to this API, applications locate resources and other program objects. A resource is a program object that provides connections to systems, such as database servers and messaging systems. (A JDBC resource is sometimes referred to as a data source.) Each resource object is identified by a unique, people-friendly name, called the JNDI name. A resource object and its JNDI name are bound together by the naming and directory service, which is included with the Application Server. To create a new resource, a new name-object binding is entered into the JNDI.

This chapter contains the following sections:

- [“Java EE Naming Services” on page 73](#)
- [“Naming References and Binding Information” on page 74](#)
- [“Using Custom Resources” on page 75](#)
- [“Using External JNDI Repositories and Resources” on page 75](#)

Java EE 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 Java EE server. Because Java EE components access this service through the JNDI API, the object usually uses its JNDI name. For example, the JNDI name of the Java DB database is `jdbc/derby`. When it starts up, the Application Server reads information from the configuration file and automatically adds JNDI database names to the name space.

Java EE 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 Java EE 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 sets and modifies 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.

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 de-couples components from resources. Because of this de-coupling, if at a later time the component needs to access a different resource, the name does not need to change. This flexibility also makes it easier for you to assemble Java EE applications from preexisting components.

The following table lists JNDI lookups and their associated references for the Java EE resources used by the Application Server.

TABLE 5-1 JNDI Lookups and Their Associated References

JNDI Lookup Name	Associated Reference
java:comp/env	Application environment entries
java:comp/env/jdbc	JDBC DataSource resource manager connection factories
java:comp/env/ejb	EJB References
java:comp/UserTransaction	UserTransaction references
java:comp/env/mail	JavaMail Session Connection Factories
java:comp/env/url	URL Connection Factories
java:comp/env/jms	JMS Connection Factories and Destinations
java:comp/ORB	ORB instance shared across application components

Using Custom Resources

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. Use the Admin Console to configure JNDI connection factory resources, J2EE resources, and access for these resources.

Using External JNDI Repositories and Resources

Often applications running on the Application Server require access to resources stored in an external JNDI repository. For example, generic Java objects could be stored in an LDAP server as per the Java schema. External JNDI resource elements let users configure such external resource repositories. The external JNDI factory must implement `javax.naming.spi.InitialContextFactory` interface.

An example of the use of an external JNDI resource is:

```
<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>
```

Connectors

A connector module (also called a resource adapter), is a Java component that enables applications to interact with enterprise information systems (EISs). EIS software includes various types of systems: enterprise resource planning (ERP), mainframe transaction processing, and non-relational databases, among others. To install a connector module you deploy it, as you do for other Java modules.

A connector connection pool is a group of reusable connections for a particular EIS. To create a connector connection pool, specify the connector module (resource adapter) that is associated with the pool.

A connector resource is a program object that provides an application with a connection to an EIS. To create a connector resource, specify its JNDI name and its associated connection pool. Multiple connector resources can specify a single connection pool. The application locates the resource by looking up its JNDI name. The JNDI name of a connector resource for an EIS is usually in the `java:comp/env/eis-specific` subcontext. Application Server 9 implements JMS by using a connector module (resource adapter)

This chapter covers:

- [“Connector Connection Pools” on page 77](#)
- [“Connector Resources” on page 79](#)
- [“Administered Object Resources” on page 79](#)

Connector Connection Pools

The following table describes connection pool settings:

Parameter	Description
Initial and Minimum Pool Size	The minimum number of connections in the pool. This value also determines the number of connections placed in the pool when the pool is first created or when application server starts.
Maximum Pool Size	The maximum number of connections in the pool.
Pool Resize Quantity	When the pool shrinks toward the minimum pool size it is resized in batches. This value determines the number of connections in the batch. Making this value too large will delay connection recycling; making it too small will be less efficient.
Idle Timeout	The maximum time in seconds that a connection can remain idle in the pool. After this time expires, the connection will be removed from the pool.
Max Wait Time	The amount of time the application that has requested a connection will wait before getting a connection time-out. Because the default wait time is long, the application might appear to hang indefinitely.
On Any Failure	If you select the checkbox labelled Close All Connections, if a single connection fails, then the application server will close all connections in the pool and then reestablish them. If you do not select the checkbox, then individual connections will be reestablished only when they are used.
Transaction Support	<p>Use the Transaction Support list to select the type of transaction support for the connection pool. The chosen transaction support overrides the transaction support attribute in the resource adapter associated with this connection pool in a downward compatible way. In other words, it can support a lower transaction level than that specified in the resource adapter or the same transaction level as that specified in resource adapter, but it cannot specify a higher level.</p> <p>The None selection from the Transaction Support menu indicates that the resource adapter does not support resource manager local or JTA transactions and does not implement <code>XAResource</code> or <code>LocalTransaction</code> interfaces. For JAXR resource adapters, you need to choose None from the Transaction Support menu. JAXR resource adapters do not support local or JTA transactions.</p> <p>Local transaction support means that the resource adapter supports local transactions by implementing the <code>LocalTransaction</code> interface. Local transactions are managed internal to a resource manager and involve no external transaction managers.</p> <p>XA transaction support means that the resource adapter supports resource manager local and JTA transactions by implementing the <code>LocalTransaction</code> and <code>XAResource</code> interfaces. XA transactions are controlled and coordinated by a transaction manager external to a resource manager. Local transactions are managed internal to a resource manager and involve no external transaction managers.</p>
Connector Validation	Select the Enabled checkbox if you want the connection pool to be validated before being passed on to the application.

Before creating a connection pool, you need to deploy the connector module (resource adapter) associated with the pool. You can deploy a connector module using the Admin Console or by using the `asadmin` command. For information about the `asadmin` command see, [asadmin\(1M\)](#)

To view, create, edit, or delete connection pools in the Admin Console, click Resources > Connectors > Connector Connection Pools. You can add properties (a name-value pair) to a connector connection pool. Alternatively, you can use the following `asadmin` commands to create and delete connection pools:

- `create-connector-connection-pool(1)`
- `delete-connector-connection-pool(1)`

Connector Resources

A connector resource provides an application with a connection to an Enterprise Information System (EIS). Every connector resource is associated with a connection pool. To view, create, edit, or delete connector resources, click Resources > Connectors > Connector Resources in the Admin Console. Alternatively, you can use the following `asadmin` commands to create and delete connection resources:

- `create-connector-resource(1)`
- `delete-connector-resource(1)`

Administered Object Resources

An administered object provides an application with specialized functionality, such as access to a parser specific to the resource adapter and its associated EIS. To view, create, edit, or delete administered objects, click Resources > Connectors > Admin Object Resources in the Admin Console.

- `create-admin-object(1)`
- `delete-admin-object-1(1)`

Java EE Containers

This chapter explains how to configure the Java EE containers included in the server. This chapter contains following sections:

- [“About the Java EE Containers” on page 81](#)
- [“Configuring Java EE 5 Containers” on page 82](#)

About the Java EE Containers

This section describes the Java EE containers included with the Application Server.

- [“Types of Java EE Containers” on page 81](#)
- [“The Web Container” on page 81](#)
- [“The EJB Container” on page 82](#)

Types of Java EE Containers

Java EE containers provide runtime support for Java EE application components. Java EE application components use the protocols and methods of the container to access other application components and services provided by the server. The Application Server provides an application client container, an applet container, a Web container, and an EJB container. For a diagram that shows the containers, see the section [“Application Server Architecture” on page 31](#).

The Web Container

The Web Container is a Java EE container that hosts web applications. The web container extends the web server functionality by providing developers the environment to run servlets and JavaServer Pages (JSP files).

The EJB Container

Enterprise beans (EJB components) are Java programming language server components that contain business logic. The EJB container provides local and remote access to enterprise beans.

There are three types of enterprise beans: session beans, entity beans, and message-driven beans. Session beans represent transient objects and processes and typically are used by a single client. Entity beans represent persistent data, typically maintained in a database. Message-driven beans are used to pass messages asynchronously to application modules and services.

The container is responsible for creating the enterprise bean, binding the enterprise bean to the naming service so other application components can access the enterprise bean, ensuring only authorized clients have access to the enterprise bean's methods, saving the bean's state to persistent storage, caching the state of the bean, and activating or passivating the bean when necessary.

Sun Java System Application Server 9 conforms to Java EE 5 and EJB 3.0 specifications. Therefore a JNDI name will no longer be required for EJB 3.0 session beans that expose a Remote view. If a deployed Remote EJB 3.0 bean does not provide a global JNDI name, the container will assign it one at runtime. The assigned JNDI name is the fully qualified class name of the remote EJB's business interface. In addition, EJB clients that use the new Remote EJB 3.0 client view will not be required to map their remote @EJB references to a global JNDI name within `sun-ejb-jar.xml`. If a JNDI name has not been associated with the `ejb` reference, the container will derive the default JNDI name for the target EJB from its associated business interface.

Applications using this EJB 3.0 feature will no longer need `sun-ejb-jar.xml` or `sun-application-client.xml` configuration files. The older method of providing a JNDI name in `sun-ejb-jar.xml` will continue to work. It will also be possible for the application to map an `ejb-reference` or `@EJB` reference to a specific global JNDI name.

See Also:

- [“Configuring the General EJB Settings” on page 85](#)

Configuring Java EE 5 Containers

- [“Configuring Web Container Sessions” on page 83](#)
- [“Configuring the General EJB Settings” on page 85](#)
- [“Configuring the Message-Driven Bean Settings” on page 87](#)
- [“Configuring the EJB Timer Service Settings” on page 87](#)

Configuring Web Container Sessions

This section describes the HTTP session settings in the Web container. HTTP sessions are unique web sessions that have their state data written to a persistent store.

- [“Configuring Session Timeout Value” on page 83](#)
- [“Configuring Manager Properties” on page 83](#)
- [“Configuring Store Properties” on page 84](#)
- [“To configure the product.name property” on page 84](#)

Configuring Session Timeout Value

Use the Admin Console to set the HTTP session timeout value. The session timeout value represents the duration for which an HTTP session is valid.

In the Admin Console, go to Configuration > Web Container > Session Properties. In the Session Timeout field, enter the number of seconds that a session is valid.

For detailed instructions on setting the session timeout value, Click Help in the Admin Console.

- [“Configuring Manager Properties” on page 83](#)
- [“Configuring Store Properties” on page 84](#)

Configuring Manager Properties

The session manager provides the means to configure how sessions are created and destroyed, where session state is stored, and the maximum number of sessions.

To change the session manager settings in the Admin Console, go to Configuration > Web Container > Manager Properties.

In the Manager Properties tab, set the following properties:

- Reap Interval value. The Reap Interval field is the number of seconds before the inactive session data is deleted from the store.
- Max Sessions value. The Max Sessions field is the maximum number of sessions allowed.
- Set the Session Filename value. The Session Filename field is the file that contains the session data.
- Session ID Generator Classname value.

The Session ID Generator Classname field allows you to specify a custom class for generating unique session IDs. Only one session ID generator class per server instance is permitted, and all instances in a cluster must use the same session ID generator to prevent session key collision.

Custom session ID generator classes must implement the `com.sun.enterprise.util.uuid.UuidGenerator` interface:

```
package com.sun.enterprise.util.uuid;

public interface UuidGenerator {

    public String generateUuid();
    public String generateUuid(Object obj); //obj is the session object
}
```

The class must be in the Application Server classpath.

For detailed instructions on setting the manager properties, Click Help in the Admin Console.

- [“Configuring Store Properties” on page 84](#)
- [“The Web Container” on page 81](#)

Configuring Store Properties

To specify where the session store data will be saved, in the Admin Console, go to Configuration > Web Container > Store Properties.

- Set the Reap Interval field
The Reap Interval field is the number of seconds before the inactive session data is deleted from the store.
- Specify the directory where session data will be stored.

For detailed instructions on setting the session store properties, Click Help in the Admin Console.

▼ To configure the product.name property

By default, the web container returns an HTTP response header with an attribute whose name is "Server" and value is the version of the Application Server Software. For example, on Application Server 9 Platform Edition, the value is: "Server: Sun Java System Application Server Enterprise Edition 9.0 FCS". Run the `asadmin version` command to view the version of your installation. If you want to override this value, define a Java Property named `product.name` in the server's configuration and initialize it to the value of your choice. This value determines what is returned by the server in the HTTP response. To change the property:

- 1 **Start the Application Server and log in to Admin Console.**
- 2 **Click Application Server and click the JVM Settings tab.**
- 3 **Click Add JVM Option.**
- 4 **Enter the following value: `-Dproduct.name="My Name"`.**

Configuring the General EJB Settings

The following section describes settings that apply to all enterprise bean containers on the server:

- “Configuring Session Store Location” on page 85
- “Configuring EJB Pool Settings” on page 85
- “Configuring EJB Cache Settings” on page 86

To override the defaults on a per-container basis, adjust the values in the enterprise bean’s `sun-ejb-jar.xml` file. For details, see the *Sun Java System Application Server Platform Edition 9 Developer’s Guide*.

Configuring Session Store Location

The Session Store Location field specifies the directory where passivated beans and persisted HTTP sessions are stored on the file system.

Passivated beans are enterprise beans that have had their state written to a file on the file system. Passivated beans typically have been idle for a certain period of time, and are not currently being accessed by clients.

Similar to passivated beans, persisted HTTP sessions are individual web sessions that have had their state written to a file on the file system.

The Commit Option field specifies how the container caches passivated entity bean instances between transactions.

Option B caches entity bean instances between transactions, and is selected by default. Option C disables caching.

Configuring EJB Pool Settings

The container maintains a pool of enterprise beans in order to respond to client requests without the performance hit that results from creating the beans. These settings only apply to stateless session beans and entity beans.

If you experience performance problems in an application that uses deployed enterprise beans, creating a pool, or increasing the number of beans maintained by an existing pool, can help increase the application’s performance.

By default, the container maintains a pool of enterprise beans.

To set the EJB pool properties using the Admin Console, go to Configuration > EJB Container > EJB Settings.

- Under Pool Settings in the Initial and Minimum Pool Size field enter the minimum number of beans the container creates in the pool.

- In the Maximum Pool Size field enter the maximum number of beans the container maintains in the pool, at any time.
- In the Pool Resize Quantity field enter the number of beans that will be removed from the pool if they are idle for more than the time specified in Pool Idle Timeout.
- In the Pool Idle Timeout field enter the time, in seconds, that a bean in the pool can remain idle before it will be removed from the pool.
- Restart the Application Server.

For detailed instructions on configuring the EJB pool, Click Help in the Admin Console.

Configuring EJB Cache Settings

The container maintains a cache of enterprise bean data for the most used enterprise beans. This allows the container to respond more quickly to requests from other application modules for data from the enterprise beans. This section applies only to stateful session beans and entity beans.

Cached enterprise beans are in one of three states: active, idle, or passivated. An active enterprise bean is currently being accessed by clients. An idle enterprise bean's data is currently in the cache, but no clients are accessing the bean. A passivated bean's data is temporarily stored, and read back into the cache if a client requests the bean.

To configure the EJB cache settings using the Admin Console, go to Configuration > EJB Container > EJB Settings.

- Adjust the maximum cache size in the Max Cache Size field.
Increase the maximum number of beans to cache to eliminate the overhead of bean creation and destruction. However, if the cache is increased, the server consumes more memory and resources. Be sure your operating environment is sufficient for your cache settings.
- Adjust the cache resize quantity in the Cache Resize Quantity field.
When the maximum number of cached beans is reached, the container removes a number of passivated beans from the backup store, set to 32 by default.
- Adjust the rate, in seconds, at which the cache cleanup is scheduled for entity beans in the Cache Idle Timeout field.
- If a cached entity bean has been idle a certain amount of time, it is passivated. That is, the bean's state is written to a backup store.
- Adjust the time, in seconds, after which stateful session beans are removed from the cache or passivated store in the Removal Timeout field.
- Configure the policy the container uses to remove stateful session beans in the Removal Selection Policy field.
The container decides which stateful session beans to remove based on the policy set in the Removal Selection Policy field. There are three possible policies the container uses to remove beans from the cache:

- Not recently used (NRU)
- First in, first out (FIFO)
- Least recently used (LRU)

The NRU policy removes a bean that hasn't been used recently. The FIFO policy removes the oldest bean in the cache. The LRU policy removes the least recently accessed bean. By default, the NRU policy is used by the container.

Entity beans are always removed using the FIFO policy.

- Restart the Application Server.

For detailed instructions on configuring the EJB pool, Click Help in the Admin Console.

Configuring the Message-Driven Bean Settings

The pool for message-driven beans is similar to the pool for session beans described in [“Configuring EJB Pool Settings” on page 85](#). By default, the container maintains a pool of message-driven beans.

To adjust the configuration of this pool using the Admin Console, go to Configuration > EJB Container > MDB Settings.

- Under Pool Settings in the Initial and Minimum Pool Size field, enter the minimum number of message beans the container creates in the pool.
- In the Maximum Pool Size field, enter the maximum number of beans the container maintains in the pool, at any time.
- In the Pool Resize Quantity field, enter the number of beans that are removed from the pool if they are idle for more than the time specified in Pool Idle Timeout.
- In the Pool Idle Timeout field, enter the time, in seconds, that a bean in the pool can remain idle before it is removed from the pool.
- Restart the Application Server.

For detailed instructions on configuring the MDB settings, Click Help in the Admin Console.

Configuring the EJB Timer Service Settings

The timer service is a persistent and transactional notification service provided by the enterprise bean container used to schedule notifications or events used by enterprise beans. All enterprise beans except stateful session beans can receive notifications from the timer service. Timers set by the service are not destroyed when the server is shut down or restarted.

To configure the EJB Timer Service using the Admin Console, go to Configuration > EJB Container > EJB Timer Service.

- Set the minimum delivery interval in milliseconds in the Minimum Delivery Interval field. Minimum Delivery Interval is the minimum number of milliseconds allowed before the next timer expiration for a particular timer can occur. Setting this interval too low can cause server overload.
- Set the maximum number of attempts the timer service makes to deliver the notification in the Maximum Redeliveries field.
- Set the interval, in milliseconds, between redelivery attempts in the Redelivery Interval field.
- Specify the JNDI name of the JDBC resource that will be used as the Timer Datasource.
- Restart the Application Server.

Using an External Database With the Timer Service

By default, the timer service uses an embedded database to store timers. You can set up the timer service to another database.

Sample timer database creation files are provided for Java DB (Derby), PointBase, Oracle, Sybase, DB2, and MS SQL Server at *install-dir/lib/install/databases/*.

To use an external database with the Timer Service, first, set up a JDBC resource for the selected database as described in [“Creating a JDBC Resource” on page 41](#). Then go to Configuration > EJB Container > EJB Timer Service and enter the JNDI name of the resource in the Timer DataSource field. Restart the Application Server.

See the following for related information:

- [“Creating a JDBC Resource” on page 41](#)
- [Chapter 5, “JNDI Resources”](#)
- [“The EJB Container” on page 82](#)

Configuring Security

This chapter describes some core application server security concepts, and describes how to configure security for the Application Server. This chapter contains the following topics:

- [“About Application Server Security” on page 89](#)
- [“Configuring Security” on page 106](#)
- [“Configuring Realms” on page 108](#)
- [“Configuring JACC Providers” on page 116](#)
- [“Configuring Audit Modules” on page 117](#)
- [“Configuring Security” on page 106](#)
- [“Configuring Listeners and JMX Connectors” on page 118](#)
- [“Configuring Connector Connection Pools” on page 123](#)
- [“Working with Certificates and SSL” on page 127](#)
- [“Further Information” on page 131](#)

About Application Server Security

- [“Overview of Security” on page 89](#)
- [“About Authentication and Authorization” on page 94](#)
- [“Understanding Users, Groups, Roles, and Realms” on page 97](#)
- [“Introduction to Certificates and SSL” on page 100](#)
- [“About Firewalls” on page 103](#)
- [“Managing Security With the Admin Console” on page 103](#)

Overview of Security

Security is about protecting data: how to prevent unauthorized access or damage to it in storage or transit. The Application Server has a dynamic, extensible security architecture based on the J2EE standard. Built in security features include cryptography, authentication and authorization, and public key infrastructure. The Application Server is built on the Java security

model, which uses a sandbox where applications can run safely, without potential risk to systems or users. The following topics are discussed:

- [“Understanding Application and System Security” on page 90](#)
- [“Tools for Managing Security” on page 90](#)
- [“Managing Security of Passwords” on page 91](#)
- [“Assigning Security Responsibilities” on page 93](#)

Understanding Application and System Security

Broadly, there are two kinds of application security:

- In *programmatic security*, application code written by the developer handles security chores. As an administrator, you don’t have any control over this mechanism. Generally, programmatic security is discouraged since it hard-codes security configurations in the application instead of managing it through the J2EE containers.
- In *declarative security*, the container (the Application Server) handles security through an application’s deployment descriptors. You can control declarative security by editing deployment descriptors directly or with a tool such as `deploytool`. Because deployment descriptors can change after an application is developed, declarative security allows for more flexibility.

In addition to application security, there is also *system security*, which affects all the applications on an Application Server system.

Programmatic security is controlled by the application developer, so this document does not discuss it; declarative security is somewhat less so, and this document touches on it occasionally. This document is intended primarily for system administrators, and so focuses on system security.

Tools for Managing Security

The Application Server provides the following tools for managing security:

- Admin Console, a browser-based tool used to configure security for the entire server, to manage users, groups, and realms, and to perform other system-wide security tasks. For a general introduction to the Admin Console, see [“Tools for Administration” on page 34](#). For an overview of the security tasks you can perform with the Admin Console, see [“Managing Security With the Admin Console” on page 103](#).
- `asadmin`, a command-line tool that performs many of the same tasks as the Admin Console. You may be able to do some things with `asadmin` that you cannot do with Admin Console. You perform `asadmin` commands from either a command prompt or through a script, to automate repetitive tasks. For a general introduction to `asadmin`, see [“Tools for Administration” on page 34](#).

The Java Enterprise Edition 5 platform (Java EE 5), provides two tools for managing security:

- `keytool`, a command-line utility for managing digital certificates and key pairs. Use `keytool` to manage users in the certificate realm.
- `policytool`, a graphical utility for managing system-wide Java security policies. As an administrator, you will rarely need to use `policytool`.

For more information on using `keytool`, `policytool`, and other Java security tools, see *Java 2 SDK Tools and Utilities* at

<http://java.sun.com/j2se/1.5.0/docs/tooldocs/index.html#security>.

Managing Security of Passwords

In this release of the Application Server, the file `domain.xml`, which contains the specifications for a particular domain, initially contains the password of the Sun Java System Message Queue broker in clear text. The element in the `domain.xml` file that contains this password is the `admin-password` attribute of the `jms-host` element. Because this password is not changeable at installation time, it is not a significant security impact.

However, use the Admin Console to add users and resources and assign passwords to these users and resources. Some of these passwords are written to the `domain.xml` file in clear text, for example, passwords for accessing a database. Having these passwords in clear text in the `domain.xml` file can present a security hazard. You can encrypt any password in `domain.xml`, including the `admin-password` attribute or a database password using the following procedure.

▼ To encrypt a password in `domain.xml`

- 1 From the directory where the `domain.xml` file resides (*domain-dir/config* by default), run the following `asadmin` command:

```
asadmin create-password-alias --user admin alias-name
```

For example,

```
asadmin create-password-alias --user admin jms-password
```

A password prompt appears (admin in this case). Refer to the man pages for the `create-password-alias`, `list-password-aliases`, `delete-password-alias` commands for more information.

- 2 Remove and replace the password in `domain.xml` using the `asadmin set` command. For example:

```
asadmin set --user admin
server.jms-service.jms-host.default_JMS_host.admin-password='${ALIAS=jms-password}'
```

- 3 Restart the Application Server for the relevant domain.

Protecting files with encoded passwords

Some files contain encoded passwords that need protecting using file system permissions. These files include the following:

- *domain-dir/master-password*
This file contains the encoded master password and should be protected with file system permission, 600.
- Any password file created to pass as an argument using the `--passwordfile` argument to `asadmin` should be protected with file system permission, 600.

▼ To change the master password

The master password (MP) is an overall shared password. It is never used for authentication and is never transmitted over the network. This password is the choke point for overall security; the user can choose to enter it manually when required, or obscure it in a file. It is the most sensitive piece of data in the system. The user can force prompting for the MP by removing this file. When the master password is changed, it is saved again in the master-password keystore.

- 1 **Stop the Application Server for the domain. Use the `asadmin change-master-password` command, which prompts for the old and new passwords, then re-encrypts all dependent items. For example,**

```
asadmin change-master-password>
Please enter the new master password>
Please enter the the new master password again>
```

- 2 **Restart the Application Server.**

▼ To change the admin password

Encrypting the admin password was discussed in [“Managing Security of Passwords” on page 91](#). Encrypting the admin password is strongly encouraged. If you want to change the admin password before encrypting it, use the `asadmin set` command. The following example shows the use of `set` command for changing the password:

```
asadmin set --user admin
server.jms-service.jms-host.default_JMS_host.admin-password=new_pwd
```

It is also possible to change the admin password using the Admin Console as in the following procedure.

- 1 **In the Admin Console tree component, expand the Configuration node.**
- 2 **Expand the Security node.**
- 3 **Expand the Realms node.**

- 4 Select the `admin-realm` node.
- 5 Click the **Manage Users** button from the **Edit Realm** page.
- 6 Select the user named `admin`.
- 7 Enter the new password and confirm the password.
- 8 Click **Save** to save or click **Close** to close without saving.

Assigning Security Responsibilities

Security responsibilities are assigned to the following:

- “Application Developer” on page 93
- “Application Deployer” on page 93
- “System Administrator” on page 93

Application Developer

The application developer is responsible for the following:

- Specifying roles and role-based access restrictions for application components.
- Defining an application’s authentication method and specifying the parts of the application that are secured.

An application developer can use tools such as NetBeans to edit application deployment descriptors. These security tasks are discussed in more detail in the *Security* chapter of *The Java EE 5 Tutorial*, which can be viewed at [Java EE 5 Tutorial](http://java.sun.com/javaee/5/docs/tutorial/doc/index.html) (<http://java.sun.com/javaee/5/docs/tutorial/doc/index.html>).

Application Deployer

The application deployer is responsible for:

- Mapping users or groups (or both) to security roles.
- Refining the privileges required to access component methods to suit the requirements of the specific deployment scenario.

System Administrator

The system administrator is responsible for:

- Configuring security realms.
- Managing user accounts and groups.
- Managing audit logs.

- **Managing server certificates and configuring the server's use of secure sockets layer (SSL).**
- Handling other miscellaneous system-wide security features, such as **security maps for connector connection pools**, additional JACC Providers, and so on.

A system administrator uses the Admin Console to manage server security settings and keytool to manage certificates. This document is intended primarily for system administrators.

About Authentication and Authorization

Authentication and authorization are central concepts of application server security. The following topics are discussed related to authentication and authorization:

- [“Authenticating Entities” on page 94](#)
- [“Authorizing Users” on page 95](#)
- [“Specifying JACC Providers” on page 95](#)
- [“Auditing Authentication and Authorization Decisions” on page 96](#)
- [“Configuring Message Security” on page 96](#)

Authenticating Entities

Authentication is the way an entity (a user, an application, or a component) determines that another entity is who it claims to be. An entity uses *security credentials* to authenticate itself. The credentials may be a user name and password or a digital certificate.

Typically, authentication means a user logging in to an application with a user name and password; but it might also refer to an EJB providing security credentials when it requests a resource from the server. Usually, servers or applications require clients to authenticate; additionally, clients can require servers to authenticate themselves, too. When authentication is bidirectional, it is called mutual authentication.

When an entity tries to access a protected resource, the Application Server uses the authentication mechanism configured for that resource to determine whether to grant access. For example, a user can enter a user name and password in a Web browser, and if the application verifies those credentials, the user is authenticated. The user is associated with this authenticated security identity for the remainder of the session.

The Application Server supports three types of authentication, as outlined in table, Authentication Methods. An application specifies the type of authentication it uses within its deployment descriptors.

TABLE 8-1 Application Server Authentication Methods

Authentication Method	Communication Protocol	Description	User Credential Encryption
Basic	HTTP (SSL optional)	Uses the server's built-in pop-up login dialog box.	None, unless using SSL.
Form-based	HTTP (SSL optional)	Application provides its own custom login and error pages.	None, unless using SSL.
Client Certificate	HTTPS (HTTP over SSL)	Server authenticates the client using a public key certificate.	SSL

For more information on using NetBeans to configure the authentication method for an application, see *Security* chapter of [Java EE 5 Tutorial](http://java.sun.com/javaee/5/docs/tutorial/doc/index.html) (<http://java.sun.com/javaee/5/docs/tutorial/doc/index.html>).

Verifying Single Sign-On

Single sign-on enables multiple applications in one virtual server instance to share user authentication state. With single sign-on, a user who logs in to one application becomes implicitly logged in to other applications that require the same authentication information.

Single sign-on is based on groups. All Web applications whose deployment descriptor defines the same *group* and use the same authentication method (basic, form, digest, certificate) share single sign-on.

Single sign-on is enabled by default for virtual servers defined for the Application Server. For information on disabling single sign-on, see “[To configure single sign-on \(SSO\)](#)” on page 121.

Authorizing Users

Once a user is authenticated, the level of *authorization* determines what operations can be performed. A user's authorization is based on his *role*. For example, a human resources application may authorize managers to view personal employee information for all employees, but allow employees to view only their own personal information. For more on roles, see “[Understanding Users, Groups, Roles, and Realms](#)” on page 97.

Specifying JACC Providers

JACC (Java Authorization Contract for Containers) is part of the Java EE 5 specification that defines an interface for pluggable authorization providers. This enables the administrator to set up third-party plug-in modules to perform authorization.

By default, the Application Server provides a simple, file-based authorization engine that complies with the JACC specification. It is also possible to specify additional third-party JACC providers.

JACC providers use the Java Authentication and Authorization Service (JAAS) APIs. JAAS enables services to authenticate and enforce access controls upon users. It implements a Java technology version of the standard Pluggable Authentication Module (PAM) framework.

Auditing Authentication and Authorization Decisions

The Application Server can provide an audit trail of all authentication and authorization decisions through *audit modules*. The Application Server provides a default audit module, as well as the ability to customize the audit modules. For information on developing custom audit modules, see the *Application Server Developer's Guide*.

Configuring Message Security

Message Security enables a server to perform end-to-end authentication of web service invocations and responses at the message layer. The Application Server implements message security using message security providers on the SOAP layer. The message security providers provide information such as the type of authentication that is required for the request and response messages. The types of authentication that are supported include the following:

- Sender authentication, including username-password authentication.
- Content authentication, including XML Digital Signatures.

Two message security providers are included with this release. The message security providers can be configured for authentication for the SOAP layer. The providers that can be configured include `ClientProvider` and `ServerProvider`.

Support for message layer security is integrated into the Application Server and its client containers in the form of (pluggable) authentication modules. By default, message layer security is disabled on the Application Server.

Message level security can be configured for the entire Application Server or for specific applications or methods. Configuring message security at the Application Server level is discussed in [Chapter 9, “Configuring Message Security.”](#) Configuring message security at the application level is discussed in [“Configuring Message Security for Web Services” in *Sun Java System Application Server Platform Edition 9 Developer's Guide*.](#)

See Also:

- [“Configuring the Application Server for Message Security” on page 140](#)
- [“Securing a Web Service” on page 138](#)
- [“To enable providers for message security” on page 144](#)
- [“To configure a message security provider” on page 145](#)
- [“Creating a Message Security Provider” on page 148](#)
- [“To delete a message security configuration” on page 151](#)
- [“To delete a message security provider” on page 151](#)
- [“To enable message security for application clients” on page 151](#)

Understanding Users, Groups, Roles, and Realms

The Application Server enforces its authentication and authorization policies upon the following entities:

- **“Users” on page 98:** An individual identity *defined in the Application Server*. In general, a user is a person, a software component such as an enterprise bean, or even a service. A user who has been authenticated is sometimes called a *principal*. Users are sometimes referred to as *subjects*.
- **“Groups” on page 98:** A set of users *defined in the Application Server*, classified by common traits.
- **“Roles” on page 98:** A named authorization level *defined by an application*. A role can be compared to a key that opens a lock. Many people might have a copy of the key. The lock doesn't care who seeks access, only that the right key is used.
- **“Realms” on page 99:** A repository containing user and group information and their associated security credentials. A realm is also called a *security policy domain*.

Note – Users and groups are designated for the entire Application Server, whereas each application defines its own roles. When the application is being packaged and deployed, the application specifies mappings between users/groups and roles, as illustrated in the following figure.

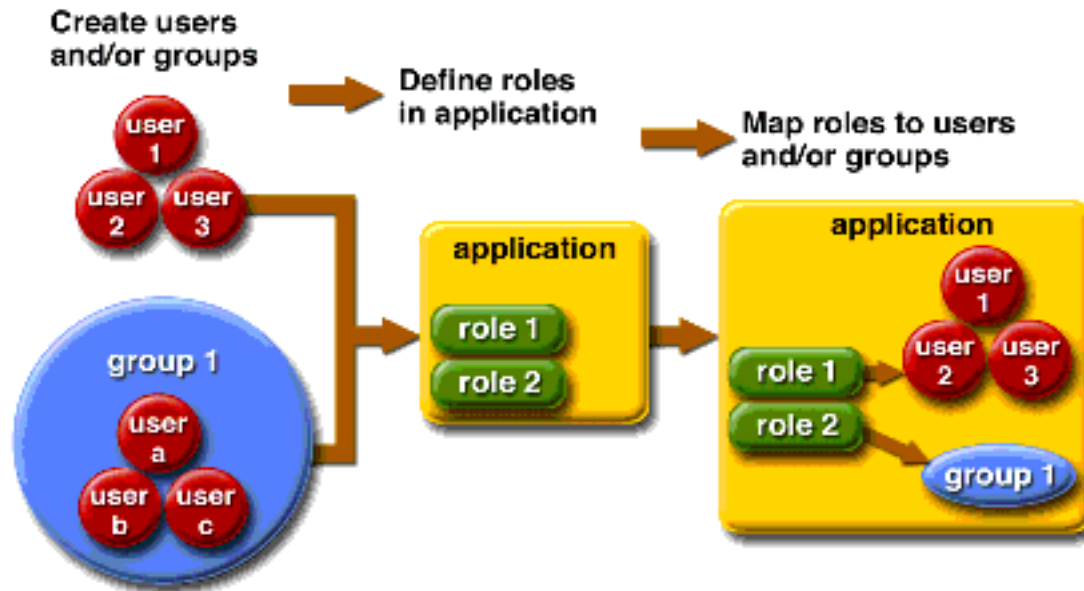


FIGURE 8-1 Role Mapping

Users

A *user* is an individual (or application program) identity that has been defined in the Application Server. A user can be associated with a group. The Application Server authentication service can govern users in multiple realms.

Groups

A *J2EE group* (or simply group) is a category of users classified by common traits, such as job title or customer profile. For example, users of an e-commerce application might belong to the customer group, but large customers would belong to the preferred group. Categorizing users into groups makes it easier to control the access of large numbers of users.

Roles

A *role* defines which applications and what parts of each application users can access and what they can do. In other words, roles determine users' authorization levels.

For example, in a personnel application all employees might have access to phone numbers and email addresses, but only managers would have access to salary information. The application might define at least two roles: employee and manager; only users in the manager role are allowed to view salary information.

A role is different from a user group in that a role defines a function in an application, while a group is a set of users who are related in some way. For example, in the personnel application there might be groups such as full-time, part-time, and on-leave, but users in all these groups would still be in the employee role.

Roles are defined in application deployment descriptors. In contrast, groups are defined for an entire server and realm. The application developer or deployer maps roles to one or more groups for each application in its deployment descriptor.

Realms

A *realm*, also called a *security policy domain* or *security domain*, is a scope over which the server defines and enforces a common security policy. In practical terms, a realm is a repository where the server stores user and group information.

The Application Server comes preconfigured with three realms: `file` (the initial default realm), `certificate`, and `admin-realm`. It is possible to also set up `ldap`, `solaris`, or `custom` realms. Applications can specify the realm to use in their deployment descriptor. If they do not specify a realm, the Application Server uses its default realm.

In the `file` realm, the server stores user credentials locally in a file named `keyfile`. You can use the Admin Console to manage users in the `file` realm. For more information, see [“Managing file Realm Users” on page 115](#).

In the `certificate` realm, the server stores user credentials in a certificate database. When using the `certificate` realm, the server uses certificates with the HTTPS protocol to authenticate Web clients. For more information about certificates, see [“Introduction to Certificates and SSL” on page 100](#).

The `admin-realm` is also a `FileRealm` and stores administrator user credentials locally in a file named `admin-keyfile`. Use the Admin Console to manage users in this realm in the same way you manage users in the `file` realm. For more information, see [“Managing file Realm Users” on page 115](#).

In the `ldap` realm the server gets user credentials from a Lightweight Directory Access Protocol (LDAP) server such as the Sun Java System Directory Server. LDAP is a protocol for enabling a user to locate organizations, individuals, and other resources such as files and devices in a network, whether on the public Internet or on a corporate intranet. Consult your LDAP server documentation for information on managing users and groups in the `ldap` realm.

In the `solaris` realm the server gets user credentials from the Solaris operating system. This realm is supported on the Solaris 9 OS and later. Consult your Solaris documentation for information on managing users and groups in the `solaris` realm.

A custom realm is any other repository of user credentials, such as a relational database or third-party component. For more information, see [“Creating a Custom Realm” on page 112](#).

Introduction to Certificates and SSL

The following topics are discussed in this section:

- [“About Digital Certificates” on page 100](#)
- [“About Secure Sockets Layer” on page 101](#)

About Digital Certificates

Digital certificates (or simply certificates) are electronic files that uniquely identify people and resources on the Internet. Certificates also enable secure, confidential communication between two entities.

There are different kinds of certificates, such as personal certificates, used by individuals, and server certificates, used to establish secure sessions between the server and clients through secure sockets layer (SSL) technology. For more information on SSL, see [“About Secure Sockets Layer” on page 101](#).

Certificates are based on *public key cryptography*, which uses pairs of digital *keys* (very long numbers) to *encrypt*, or encode, information so it can be read only by its intended recipient. The recipient then *decrypts* (decodes) the information to read it.

A key pair contains a public key and a private key. The owner distributes the public key and makes it available to anyone. But the owner never distributes the private key; it is always kept secret. Because the keys are mathematically related, data encrypted with one key can be decrypted only with the other key in the pair.

A certificate is like a passport: it identifies the holder and provides other important information. Certificates are issued by a trusted third party called a *Certification Authority* (CA). The CA is analogous to a passport office: it validates the certificate holder's identity and signs the certificate so that it cannot be forged or tampered with. Once a CA has signed a certificate, the holder can present it as proof of identity and to establish encrypted, confidential communications.

Most importantly, a certificate binds the owner's public key to the owner's identity. Like a passport binds a photograph to personal information about its holder, a certificate binds a public key to information about its owner.

In addition to the public key, a certificate typically includes information such as:

- The name of the holder and other identification, such as the URL of the Web server using the certificate, or an individual's email address.
- The name of the CA that issued the certificate.
- An expiration date.

Digital Certificates are governed by the technical specifications of the X.509 format. To verify the identity of a user in the certificate realm, the authentication service verifies an X.509 certificate, using the common name field of the X.509 certificate as the principal name.

About Certificate Chains

Web browsers are preconfigured with a set of *root* CA certificates that the browser automatically trusts. Any certificates from elsewhere must come with a *certificate chain* to verify their validity. A certificate chain is series of certificates issued by successive CA certificates, eventually ending in a root CA certificate.

When a certificate is first generated, it is a *self-signed* certificate. A self-signed certificate is one for which the issuer (signer) is the same as the subject (the entity whose public key is being authenticated by the certificate). When the owner sends a certificate signing request (CSR) to a CA, then imports the response, the self-signed certificate is replaced by a chain of certificates. At the bottom of the chain is the certificate (reply) issued by the CA authenticating the subject's public key. The next certificate in the chain is one that authenticates the CA's public key. Usually, this is a self-signed certificate (that is, a certificate from the CA authenticating its own public key) and the last certificate in the chain.

In other cases, the CA can return a chain of certificates. In this case, the bottom certificate in the chain is the same (a certificate signed by the CA, authenticating the public key of the key entry), but the second certificate in the chain is a certificate signed by a different CA, authenticating the public key of the CA to which you sent the CSR. Then, the next certificate in the chain is a certificate authenticating the second CA's key, and so on, until a self-signed *root* certificate is reached. Each certificate in the chain (after the first) thus authenticates the public key of the signer of the previous certificate in the chain.

About Secure Sockets Layer

Secure Sockets Layer (SSL) is the most popular standard for securing Internet communications and transactions. Web applications use HTTPS (HTTP over SSL), which uses digital certificates to ensure secure, confidential communications between server and clients. In an SSL connection, both the client and the server encrypt data before sending it, then decrypt it upon receipt.

When a Web browser (client) wants to connect to a secure site, an *SSL handshake* is initiated, as described in the following procedure:

- The browser sends a message over the network requesting a secure session (typically, by requesting a URL that begins with `https` instead of `http`).
- The server responds by sending its certificate (including its public key).
- The browser verifies that the server's certificate is valid and is signed by a CA whose certificate is in the browser's database (and who is trusted). It also verifies that the CA certificate has not expired.
- If the certificate is valid, the browser generates a one time, unique *session key* and encrypts it with the server's public key. The browser then sends the encrypted session key to the server so that they both have a copy.
- The server decrypts the message using its private key and recovers the session key.

After the handshake, the client has verified the identity of the Web site, and only the client and the Web server have a copy of the session key. From this point forward, the client and the server use the session key to encrypt all their communications with each other. Thus, their communications are ensured to be secure.

The newest version of the SSL standard is called TLS (Transport Layer Security). The Application Server supports the Secure Sockets Layer (SSL) 3.0 and the Transport Layer Security (TLS) 1.0 encryption protocols.

To use SSL, the Application Server must have a certificate for each external interface, or IP address, that accepts secure connections. The HTTPS service of most Web servers will not run unless a digital certificate has been installed. Use the procedure described in [“To generate a certificate using the keytool utility” on page 130](#) to set up a digital certificate that your Web server can use for SSL.

About Ciphers

A *cipher* is a cryptographic algorithm used for encryption or decryption. SSL and TLS protocols support a variety of ciphers used to authenticate the server and client to each other, transmit certificates, and establish session keys.

Some ciphers are stronger and more secure than others. Clients and servers can support different cipher suites. Choose ciphers from the SSL3 and TLS protocols. During a secure connection, the client and the server agree to use the strongest cipher they both have enabled for communication, so it is usually sufficient to enable all ciphers.

Using Name-based Virtual Hosts

Using name-based virtual hosts for a secure application can be problematic. This is a design limitation of the SSL protocol itself. The SSL handshake, where the client browser accepts the server certificate, must occur before the HTTP request is accessed. As a result, the request information containing the virtual host name cannot be determined prior to authentication, and it is therefore not possible to assign multiple certificates to a single IP address.

If all virtual hosts on a single IP address need to authenticate against the same certificate, the addition of multiple virtual hosts probably will not interfere with normal SSL operations on the server. Be aware, however, that most browsers will compare the server's domain name against the domain name listed in the certificate, if any (applicable primarily to official, CA-signed certificates). If the domain names do not match, these browsers display a warning. In general, only address-based virtual hosts are commonly used with SSL in a production environment.

About Firewalls

A *firewall* controls the flow of data between two or more networks, and manages the links between the networks. A firewall can consist of both hardware and software elements. This section describes some common firewall architectures and their configuration. The information here pertains primarily to the Application Server. For details about a specific firewall technology, refer to the documentation from your firewall vendor.

In general, configure the firewalls so that clients can access the necessary TCP/IP ports. For example, if the HTTP listener is operating on port 8080, configure the firewall to allow HTTP requests on port 8080 only. Likewise, if HTTPS requests are setup for port 8181, you must configure the firewalls to allow HTTPS requests on port 8181.

If direct Remote Method Invocations over Internet Inter-ORB Protocol (RMI-IIOP) access from the Internet to EJB modules are required, open the RMI-IIOP listener port as well, but this is strongly discouraged because it creates security risks.

In double firewall architecture, you must configure the outer firewall to allow for HTTP and HTTPS transactions. You must configure the inner firewall to allow the HTTP server plug-in to communicate with the Application Server behind the firewall.

Managing Security With the Admin Console

The Admin Console provides the means to manage the following aspects of security:

- “[Server Security Settings](#)” on page 103
- “[Realms and file Realm Users](#)” on page 103
- “[JACC Providers](#)” on page 104
- “[Audit Modules](#)” on page 104
- “[Message Security](#)” on page 104
- “[HTTP and IIOP Listener Security](#)” on page 105
- “[Admin Service Security](#)” on page 105
- “[Security Maps](#)” on page 106

Server Security Settings

On the Security Settings page, set properties for the entire server, including specifying the default realm, the anonymous role, and the default principal user name and password. For more information, see “[Configuring General Security Settings](#)” on page 106.

Realms and file Realm Users

The concept of realms was introduced in “[Understanding Users, Groups, Roles, and Realms](#)” on page 97.

See “[Configuring Realms](#)” on page 108 for details on these tasks:

- [“Creating a Realm” on page 108](#)
- [“Editing a Realm” on page 109](#)
- [“Setting the Default Realm” on page 110](#)

JACC Providers

JACC providers were introduced in [“Specifying JACC Providers” on page 95](#). Use the Admin Console to perform the following tasks:

- Add a new JACC provider
- Delete or modify an existing JACC provider

See [“Configuring JACC Providers” on page 116](#) for details on these tasks.

See Also:

- [“Creating a JACC Provider” on page 116](#)
- [“Setting the Active JACC Provider” on page 117](#)

Audit Modules

Audit modules were introduced in [“Auditing Authentication and Authorization Decisions” on page 96](#). Auditing is the method by which significant events, such as errors or security breaches, are recorded for subsequent examination. All authentication events are logged to the Application Server logs. A complete access log provides a sequential trail of Application Server access events.

Use the Admin Console to perform the following tasks:

- Add a new audit module
- Delete or modify an existing audit module

See [“Configuring Audit Modules” on page 117](#) for details on these tasks.

See Also:

- [“Creating an Audit Module” on page 117](#)
- [“Enabling or Disabling Audit Logging” on page 118](#)

Message Security

The concept of message security was introduced in [“Configuring Message Security” on page 96](#). Use the Admin Console to perform the following tasks:

- Enable message security
- Configure a message security provider
- Delete or configure an existing message security configuration or provider

See [Chapter 9, “Configuring Message Security,”](#) for details on these tasks.

See Also:

- [“Configuring the Application Server for Message Security” on page 140](#)
- [“Securing a Web Service” on page 138](#)
- [“To enable providers for message security” on page 144](#)
- [“To configure a message security provider” on page 145](#)
- [“Creating a Message Security Provider” on page 148](#)
- [“To delete a message security configuration” on page 151](#)
- [“To delete a message security provider” on page 151](#)
- [“To enable message security for application clients” on page 151](#)

HTTP and IIOP Listener Security

Each virtual server in the HTTP service provides network connections through one or more *HTTP listeners*. For general information about the HTTP service and HTTP listeners, see [Chapter 11, “Configuring the HTTP Service”](#)

The Application Server supports CORBA (Common Object Request Broker Architecture) objects, which use the Internet Inter-Orb Protocol (IIOP) to communicate across the network. An *IIOP listener* accepts incoming connections from remote clients of EJB components and from other CORBA-based clients. For general information on IIOP listeners, see [“IIOP Listeners” on page 178](#).

With the Admin Console, perform the following tasks:

- Create a new HTTP or IIOP listener, and specify the security it uses.
- Modify the security settings for an existing HTTP or IIOP listener.

See [“Configuring Listeners and JMX Connectors” on page 118](#) for details on these tasks.

See Also:

- [“Configuring Security for HTTP Listeners” on page 118](#)
- [“Configuring Security for IIOP Listeners” on page 118](#)
- [“To set listener security properties” on page 119](#)

Admin Service Security

The Admin Service determines whether the server instance is a regular instance, a domain administration server (DAS), or a combination. In the Platform Edition, there is only one server instance, and it is a combination. Use the Admin Service to configure a JSR-160 compliant remote JMX connector, which handles communication between the domain administration server and the node agents, which manage server instances on a host machine, for remote server instances.

With the Admin Console, perform the following tasks:

- Manage the Admin Service

- Edit the JMX connector
- Modify the security settings of the JMX connector

See [“Configuring Security For The Admin Service’s JMX Connector” on page 119](#) for details on these tasks.

See Also:

- [“Configuring Security For The Admin Service’s JMX Connector” on page 119](#)

Security Maps

The concept of security maps for connector connection pools is introduced in [“About Security Maps” on page 123](#). Use the Admin Console to perform the following tasks:

- Add a security map to an existing connector connection pool
- Delete or configure an existing security map

See [“Configuring Connector Connection Pools” on page 123](#) for details on these tasks.

See Also:

- [“About Connector Connection Pools” on page 123](#)
- [“About Security Maps” on page 123](#)
- [“To create a security map” on page 124](#)
- [“To edit a security map” on page 125](#)
- [“To delete a security map” on page 126](#)

Configuring Security

This section contains the following topics:

- [“Configuring General Security Settings” on page 106](#)
- [“Granting Access to Administration Tools” on page 108](#)
- [“Configuring Mutual Authentication” on page 115](#)
- [“To configure single sign-on \(SSO\)” on page 121](#)

Configuring General Security Settings

Use the Security page in the Admin Console to set a variety of system-wide security settings.

Go to Configuration > Security. The Security page displays with general security options. These options are summarized in the following table.

Setting	Description
Security Manager	<p>Select the Enable checkbox to turn on the security manager for the domain.</p> <p>When enabled, a JVM option, <code>-Djava.security.manager</code>, will be added to the JVM setting of the Application Server. You must restart the server to enable this change.</p> <p>Ensure that you have granted correct permissions for all applications. You can turn off the security manager to enhance performance.</p>
Audit Logging	Select to enable audit logging. If enabled, the server will load and run all the audit modules specified in the Audit Modules setting. If disabled, the server does not access audit modules. Disabled by default.
Default Realm	The active (default) realm the server uses for authentication. Applications use this realm unless they specify a different realm in their deployment descriptor. All configured realms appear in the list. The initial default realm is the <code>file</code> realm.
Anonymous Role	The name for the default or anonymous role. The anonymous role is assigned to all users. Applications can use this role in their deployment descriptors to grant authorization to anyone.
Default Principal	<p>Specifies the default user name. The server uses this when no principal is provided. If you enter a value in this field, enter a corresponding value in the Default Principal Password field.</p> <p>This attribute is not required for normal server operation.</p>
Default Principal Password	<p>Password of the default principal specified in the Default Principal field.</p> <p>This attribute is not required for normal server operation.</p>
JACC	Class name of a configured JACC provider. See “Creating a JACC Provider” on page 116
Audit Modules	List of audit module provider classes, delimited by commas. A module listed here must already be configured. If Audit Logging is enabled, this setting must list audit modules. By default, the server uses an audit module named <code>default</code> . For information on creating new audit modules, see “Creating an Audit Module” on page 117 .
Default Principal To Role Mapping	Check to apply a default principal-to-role mapping to applications that do not have an application-specific mapping.
Mapped Principal Classes	Customize the <code>java.security.Principal</code> implementation class used in the default principal-to-role mapping.

For more details on configuring all the options on the Security page, click Help in the Admin Console.

Granting Access to Administration Tools

Only users in the `asadmin` group are able to access Admin Console and the `asadmin` command line utility.

To give a user access to these administration tools, add them to the `asadmin` group in the `admin-realm`. In the Admin Console, go to Configuration > Security > Realms > `admin-realm` > Edit Realm > Manage Users. If the user name exists, click on the user name to edit settings or click New to add a new user name.

Initially after installation, the administrator user name and password entered during installation are listed in a file named `admin-keyfile`. By default, this user belongs to the group `asadmin`, which gives rights to modify the Application Server. Assign users to this group only if you want to grant them administrator privileges for the Application Server.

If you add users to the `admin-realm` realm, but assign the user to a group other than `asadmin`, the user information will still be written to the file named `admin-keyfile`, but the user will have no access to administrative tools or to applications in the `file` realm.

To authorize a user to make modifications to the Application Server, include the `asadmin` group in the Group List.

For detailed instructions on setting up a new user account with admin privileges, click Help in the Admin Console.

See Also:

- [“Editing a Realm” on page 109](#)
- [“Setting the Default Realm” on page 110](#)
- [“Realms and file Realm Users” on page 103](#)

Configuring Realms

This section contains the following topics:

- [“Creating a Realm” on page 108](#)
- [“Editing a Realm” on page 109](#)
- [“Setting the Default Realm” on page 110](#)
- [“Additional Information for Specific Realms” on page 110](#)

Creating a Realm

The Application Server comes preconfigured with three realms: `file`, `certificate`, and `admin-realm`. It is also possible to create `ldap`, `solaris`, and custom realms. Generally, you will

have one realm of each type on a server, but on the Application Server there are two file realms: `file` and `admin-realm`. These are two realms of the same type used for two different purposes. It is also possible to have a different certificate database for each virtual server on your system.

To create a realm using the Admin Console, go to Configuration > Security > Realms > New. Enter a name for the realm and specify the class name for the realm you are creating. Class names for different realms are shown in the following table:

Realm Name	Class Name
<code>file</code>	<code>com.sun.enterprise.security.auth.realm.file.FileRealm</code>
<code>certificate</code>	<code>com.sun.enterprise.security.auth.realm.certificate.CertificateRealm</code>
<code>ldap</code>	<code>com.sun.enterprise.security.auth.realm.ldap.LDAPRealm</code>
<code>solaris</code>	<code>com.sun.enterprise.security.auth.realm.solaris.SolarisRealm</code>
<code>custom</code>	Name of login realm class

Add the required properties and any desired optional properties for the realm.

- For a description of file realm properties, see [“Editing the file and admin-realm Realms” on page 114](#).
- For a description of certificate realm properties, see [“Editing the certificate Realm” on page 113](#).
- For a description of ldap realm properties, see [“Creating an ldap Realm” on page 110](#).
- For a description of solaris realm properties, see [“Creating the solaris Realm” on page 112](#).
- For a description of custom realm properties, see [“Creating a Custom Realm” on page 112](#).

For more details on creating a realm, click Help in the Admin Console.

Equivalent `asadmin` command.

Use the `create-auth-realm` `asadmin` command to create a realm. For details, see [`create-auth-realm\(1\)`](#).

Editing a Realm

In the Admin Console, go to Configuration > Security > Realms and select the realm you want to edit. The Edit Realm page will display, where you can make changes to the current configuration.

For details on editing or deleting a realm, click Help in the Admin Console.

Equivalent `asadmin` command.

Use the `delete-auth-realm` command to delete a realm. For details, [`delete-auth-realm\(1\)`](#).

Setting the Default Realm

The *default realm* is the realm that the Application Server uses for authentication and authorization if an application’s deployment descriptor does not specify a realm.

In the Admin Console, go to Configuration > Security and select the required realm from the Default Realm drop-down list. Click Save.

For more details, click Help in the Admin Console.

Additional Information for Specific Realms

This section covers the following topics:

- “Creating an ldap Realm” on page 110
- “Creating the solaris Realm” on page 112
- “Creating a Custom Realm” on page 112
- “Editing the certificate Realm” on page 113
- “Editing the file and admin-realm Realms” on page 114
- “Managing Users with Network Security Services (NSS)” on page 114
- “Managing file Realm Users” on page 115
- “Configuring Mutual Authentication” on page 115

Creating an ldap Realm

The ldap realm performs authentication using information from an LDAP server. User information includes user name, password, and the groups to which the user belongs. To use an LDAP realm, the users and groups must already be defined in your LDAP directory.

To create an LDAP realm, follow the steps in “Creating a Realm” on page 108 for adding a new realm, and add the properties shown in the following table.

TABLE 8-2 Required properties for ldap realm

Property Name	Description	Value
directory	LDAP URL of the directory server.	LDAP URL of the form ldap://hostname:portFor example, ldap://myldap.foo.com:389.
base-dn	Base Distinguished Name (DN) for the location of user data, which can be at any level above the user data, since a tree scope search is performed. The smaller the search tree, the better the performance.	Domain for the search, for example: dc=siliconvalley, dc=BayArea, dc=sun, dc=com.

TABLE 8-2 Required properties for ldap realm (Continued)

Property Name	Description	Value
jaas-context	Type of login module to use for this realm.	Must be <code>LdapRealm</code> .

Optional properties for the ldap realm are shown in the following table.

TABLE 8-3 Optional properties for ldap realm

Property Name	Description	Default
search-filter	Search filter to use to find the user.	<code>uid=%s</code> (%s expands to the subject name).
group-base-dn	Base DN for the location of group data.	Same as the <code>base-dn</code> , but it can be tuned if necessary.
group-search-filter	Search filter to find group memberships for the user.	<code>uniquemember=%d</code> (%d expands to the user element DN).
group-target	LDAP attribute name that contains group name entries.	CN
search-bind-dn	Optional DN used to authenticate to the directory for performing the search-filter lookup. Only required for directories that do not allow anonymous search.	
search-bind-password	LDAP password for the DN given in <code>search-bind-dn</code> .	

Example

For example, suppose an LDAP user, Joe Java, is defined in the LDAP directory as follows:

```
uid=jjava,ou=People,dc=acme,dc=com
uid=jjava
givenName=joe
objectClass=top
objectClass=person
objectClass=organizationalPerson
objectClass=inetorgperson
sn=java
cn=Joe Java
```

Using the example code, when creating or editing the ldap realm, you can enter the values as shown in the following table.

TABLE 8-4 Example ldap realm values

Property Name	Property Value
directory	LDAP URL to your server, for example: <code>ldap://ldap.acme.com:389</code>
base-dn	<code>ou=People,dc=acme,dc=com</code> . Can be rooted higher, for example <code>dc=acme,dc=com</code> , but searches would traverse a larger part of the tree, reducing performance.
jaas-context	<code>ldapRealm</code>

Creating the solaris Realm

The `solaris` realm gets user and group information from the underlying Solaris user database, as determined by the system’s configuration. The `solaris` realm invokes the underlying PAM infrastructure for authenticating. If the configured PAM modules require root privileges, the domain must run as root to use this realm. For details, see the Solaris documentation for security services.

The `solaris` realm has one required property, `jaas-context` that specifies the type of login module to use. The property value must be `solarisRealm`.

Note – The `solaris` realm is supported only for Solaris 9 or later.

Creating a Custom Realm

In addition to the four built-in realms, you can also create custom realms that store user data in some other way, such as in a relational database. Development of a custom realm is outside the scope of this document. For more information, see [Chapter 5, “Securing Applications,” in *Sun Java System Application Server Platform Edition 9 Developer’s Guide*](#) in the *Application Server Developer’s Guide*.

As an administrator, the main thing you need to know is that a custom realm is implemented by a class (called the `LoginModule`) derived from the Java Authentication and Authorization Service (JAAS) package.

▼ To create a custom realm

- 1 Follow the procedure outline in [“Creating a Realm” on page 108](#), entering the name of the custom realm and the name of the `LoginModule` class.
Any unique name can be used for the custom realm, for example `myCustomRealm`.
- 2 Add the properties for a custom realm shown in the following table.

Property Name	Property Value
jaas-context	LoginModule class name, for example <code>simpleCustomRealm</code>
auth-type	Description of the realm, for example “A simple example custom realm”.

3 Click OK.

4 Edit the domain's login configuration file, *domain-dir/config/login.conf*, and add the fully-qualified class name of the JAAS LoginModule at the end of the file, as follows:

```
realmName {
    fully-qualified-LoginModule-classname required;
};
```

For example,

```
myCustomRealm {
    com.foo.bar.security.customrealm.simpleCustomLoginModule required;
};
```

5 Copy the LoginModule class and all dependent classes into the directory *domain-dir/lib/classes*.

6 Restart the Server if Restart Required displays in the console.

7 Make sure that the realm is properly loaded.

Check *domain-dir/logs/server.log* to make sure the server loaded the realm. The server should invoke the realm's `init()` method.

Editing the certificate Realm

The certificate realm supports SSL authentication. This realm sets up the user identity in the Application Server's security context, and populates it with user data obtained from cryptographically verified client certificates in the truststore and keystore files (see “[About Certificate Files](#)” on page 127). Add users to these files using `keytool`. For more information, see chapter titled *Security* at [Java EE 5 Tutorial](#) (<http://java.sun.com/javaee/5/docs/tutorial/doc/index.html>).

With the certificate realm, Java EE containers handle authorization processing based on each user's Distinguished Name (DN) from his or her certificate. The DN is the name of the entity whose public key the certificate identifies. This name uses the X.500 standard, so it is intended to be unique across the Internet. For more information on key stores and trust stores, refer to the `keytool` documentation at <http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html>.

The following table lists the optional properties for the certificate realm.

TABLE 8-5 Optional properties for certificate realm

Property	Description
assign-groups	A comma-separated list of group names. All clients who present valid certificates are assigned to these groups. For example, <code>employee,manager</code> , where these are the names of user groups.
jaas-context	Type of login module to use for this realm. For the certificate realm, the value must be <code>certificateRealm</code> .

Editing the file and admin-realm Realms

The server maintains all user, group, and password information in a file named `keyfile` for the file realm and `admin-keyfile` for the admin-realm. For both, the `file` property specifies the location of the keyfile. The following table shows required properties for a file realm.

TABLE 8-6 Required properties for file realms

Property name	Description	Default Value
file	Full path and name of the keyfile.	<code>domain-dir/config/keyfile</code>
jaas-context	Type of login module to use for this realm.	<code>fileRealm</code> is the only valid value

The `keyfile` is initially empty, so users must be added before the file realm is used. For instructions, see [“Managing file Realm Users” on page 115](#).

The `admin-keyfile` initially contains the admin user name, the admin password in an encrypted format, and the group to which this user belongs, which is `asadmin` by default. For more information on adding users to the admin-realm, read [“Granting Access to Administration Tools” on page 108](#).

Note – Users in the group `asadmin` in the admin-realm are authorized to use the Admin Console and `asadmin` tools. Add only users to this group that have server administrative privileges.

Managing Users with Network Security Services (NSS)

In the **Enterprise Edition only**, you can manage users using the Admin Console as discussed in [“Managing file Realm Users” on page 115](#) or you can manage users using NSS tools. Network Security Services (NSS) is a set of libraries designed to support cross-platform development of security-enabled client and server applications. Applications built with NSS can support SSL v2 and v3, TLS, PKCS #5, PKCS #7, PKCS #11, PKCS #12, S/MIME, X.509 v3 certificates, and other security standards. For detailed information, link to the following URLs:

- Network Security Services (NSS) at <http://www.mozilla.org/projects/security/pki/nss/>

- NSS Security Tools at <http://www.mozilla.org/projects/security/pki/nss/tools/>
- Overview of NSS at <http://www.mozilla.org/projects/security/pki/nss/overview.html>

Managing file Realm Users

Manage file realm users with the Admin Console. Users and groups in the file realm are listed in the keyfile, whose location is specified by the file property.

Note – It is also possible to use these steps to add users to any file realm, including the admin-realm. Simply substitute the name of the target realm in place of the file realm referenced in this section.

A user in the file realm can belong to a *JavaEE group*, a category of users classified by common traits. For example, customers of an e-commerce application might belong to the CUSTOMER group, but the big spenders would belong to the PREFERRED group. Categorizing users into groups makes it easier to control the access of large numbers of users.

Initially after installation of the Application Server, the only user is the administrator entered during installation. By default, this user belongs to the group asadmin, in the realm admin-realm, which gives rights to modify the Application Server. Any users assigned to this group will have administrator privileges, that is, they will have access to the asadmin tool and the Admin Console.

To manage file realm users, perform the following tasks:

- Add a user.
- Edit user information.
- Delete a user.

To access the File Users page in the Admin Console, go to Configuration > Security > Realms > File > Edit Realm > Manage Users. For detailed information on performing these tasks, click Help in the Admin Console.

Equivalent asadmin commands.

The same tasks can be performed from the asadmin command line utility. For more information, see

- create-file-user: See [create-file-user\(1\)](#).
- update-file-user: See [update-file-user\(1\)](#).
- delete-file-user: See [delete-file-user\(1\)](#).

Configuring Mutual Authentication

- “Enabling Mutual SSL Authentication in an Application” on page 116
- “Enabling Mutual Authentication For All Applications” on page 116

In mutual authentication, both server and client-side authentication are enabled. To test mutual authentication, a client with a valid certificate must exist. For information on mutual authentication, see the *Security* chapter of [Java EE 5 Tutorial](http://java.sun.com/javaee/5/docs/tutorial/doc/index.html) (<http://java.sun.com/javaee/5/docs/tutorial/doc/index.html>).

Enabling Mutual SSL Authentication in an Application

To enable mutual authentication for a specific application, use NetBeans to set the method of authentication to `Client-Certificate`. For more information on using NetBeans, see the [Java EE 5 Tutorial](http://java.sun.com/javaee/5/docs/tutorial/doc/index.html) (<http://java.sun.com/javaee/5/docs/tutorial/doc/index.html>).

Enabling Mutual Authentication For All Applications

The Application Server uses the `certificate` realm for HTTPS authentication. To enforce client authentication for all applications that use the `certificate` realm, in the Admin Console, go to `Configuration > Security > Realms > certificate realm`. Click `Add Property` and enter the following values and click `Save`:

- In the `Name` field, enter `clientAuth`.
- In the `Value` field, enter `true`.

Restart the Application Server if `Restart Required` displays in the console.

Configuring JACC Providers

JACC (Java Authorization Contract for Containers) is part of the Java EE 5 specification that defines an interface for pluggable authorization providers. This enables the administrator to set up third-party *plug in* modules to perform authorization. By default, the Application Server provides a simple, JACC-compliant, file-based authorization engine.

- [“Creating a JACC Provider” on page 116](#)
- [“Setting the Active JACC Provider” on page 117](#)

Creating a JACC Provider

In the Admin Console, go to `Configuration > Security > JACC Providers`. Click `New` and enter the required values.

To edit or delete a JACC provider, go to `Configuration > Security > JACC Providers` and select the provider you want to edit or delete.

For details on creating, editing or deleting JACC providers, click `Help` in the Admin Console.

Setting the Active JACC Provider

In the Admin Console, go to Configuration > Security. In the JACC field enter the name of the JAAC provider to be used by the server. Click Save.

For details, click Help in the Admin Console.

Configuring Audit Modules

The audit module logs authentication and authorization requests to the server log file. For information on changing the location of the log file, see [“Configuring General Logging Settings” on page 186](#).

Authentication log entries include the following information:

- Names of users who attempted to authenticate.
- The realm that processed the access request.
- The requested Web module URI or EJB component.
- Success or failure of the request.

Regardless of whether audit logging is enabled, the Application Server logs all denied authentication events.

Authorization log entries include the following information:

- Names of authenticated users, if any.
- The requested Web URI or EJB component.
- Success or failure of the requests.
- [“Creating an Audit Module” on page 117](#)
- [“Setting the Active Audit Module” on page 118](#)
- [“Enabling or Disabling Audit Logging” on page 118](#)

Creating an Audit Module

In the Admin Console, go to Configuration > Security > Audit Modules. In the Audit modules page click New and enter the required values. Click OK.

To edit or delete an audit module, go to Configuration > Security > Audit Modules, and select an audit module. In the Edit Audit Module page, modify your settings and click Save. To delete, select the check box to the left of an audit module, and click Delete.

For detailed information on managing audit modules, click Help in the Admin Console.

Enabling or Disabling Audit Logging

Logging for audit modules is not turned on by default. To enable logging, in the Admin Console, go to Configuration > Security. Select the Audit Logging check box to enable logging. Deselect to disable logging.

For detailed information on enabling or disabling audit logging, click Help in the Admin Console.

Setting the Active Audit Module

To specify the audit module that the server uses, first enable audit logging as described in [“Enabling or Disabling Audit Logging” on page 118](#). In the Audit Modules field, enter the name of the audit module to be used by the server.

The preconfigured audit module is called `default`. Make sure that this audit module has `auditOn`.

For details, click Help in the Admin Console.

Configuring Listeners and JMX Connectors

- [“Configuring Security for HTTP Listeners” on page 118](#)
- [“Configuring Security for IIOP Listeners” on page 118](#)
- [“Configuring Security For The Admin Service’s JMX Connector” on page 119](#)
- [“To set listener security properties” on page 119](#)

Configuring Security for HTTP Listeners

Each virtual server in the HTTP service provides network connections through one or more *HTTP listeners*. To configure security for HTTP listeners using the Admin Console, go to Configuration > HTTP Service > HTTP Listeners. Select an HTTP listener to modify its settings. Follow the procedure in [“To set listener security properties” on page 119](#) to set security properties.

Equivalent `asadmin` command: `create-http-listener`.

Configuring Security for IIOP Listeners

The Application Server supports CORBA (Common Object Request Broker Architecture) objects, which use the Internet Inter-Orb Protocol (IIOP) to communicate across the network.

An *IIOP listener* accepts incoming connections from remote clients of EJB components and from other CORBA-based clients. With the Admin Console, create new IIOP listeners and edit the settings of existing IIOP listeners.

To configure security for IIOP listeners, in the Admin Console, go to Configuration > ORB > IIOP Listeners. Follow the procedure in [“To set listener security properties” on page 119](#) to set security properties.

Equivalent `asadmin` command: `create-iiop-listener`.

Configuring Security For The Admin Service’s JMX Connector

In the Admin Console, go to Configuration > Admin Service. Select the admin service to modify. Follow the procedure in [“To set listener security properties” on page 119](#) to set security properties.

▼ To set listener security properties

This procedure applies to HTTP listener, IIOP listener, and JMX Connector security properties.

- 1 In the Edit HTTP Listener, Edit IIOP Listener, or Edit JMX Connector page, go to the section labeled SSL.
- 2 Check the Enabled box in the Security field to enable security for this listener. When this option is selected, you must select SSL3 or TLS to specify which type of security is enabled, and you must enter a certificate nickname.
- 3 Check the Enabled box in the Client Authentication field if clients are to authenticate themselves to the Application Server when using this listener.
- 4 Enter the keystore alias in the Certificate Nickname field if the Enabled box is checked. The keystore alias is a single value that identifies an existing server key pair and certificate. The certificate nickname for the default keystore is `slas`.

To find the Certificate Nickname, use `keytool`, as shown in the following example:
`keytool -list -v -keystore keystore.jks`

If the name has changed in the keystore file, then use that name instead of `keystore.jks`.

- 5 Select SSL3 and/or TLS if the Enabled box is checked. By default, both SSL3 and TLS are enabled.
- 6 Enable individual cipher suites, if needed. By default, all supported cipher suites are enabled. Ciphers are discussed in [“About Ciphers” on page 102](#).

7 Select Save to save the changes or Load Defaults to cancel.

- See Also**
- [“Configuring Security for IIOP Listeners” on page 118](#)
 - [“Configuring Security for HTTP Listeners” on page 118](#)
 - [“Configuring Security For The Admin Service’s JMX Connector” on page 119](#)

▼ To secure CORBA objects

CORBA objects include Java RMI-IIOP and Java IDL or POA based CORBA objects, excluding EJB modules. By default, authentication is not required for CORBA objects.

1 Configure authentication, if desired.

- Expand the ORB node, and then the IIOP Listeners node.**
- Click the SSL listener.**
- Select the Security Enabled checkbox.**
- Click Save.**
- Restart the Application Server.**

Once authentication is turned on, all clients need to authenticate by supplying a user name and password (if using basic authentication) or a certificate (if using SSL mutual authentication).

2 Configure authorization.

To turn on authorization for CORBA objects, specify the appropriate security policy in the server’s security configuration file, *domain-dir/config/server.policy*.

By default, all users are allowed to access all non-EJB CORBA objects in the server, as specified by the following default grant block:

```
grant { permission com.sun.enterprise.security.CORBAObjectPermission "*", "*"; }
```

CORBAObjectPermission is a special Java Permission class that controls which users are allowed to access non-EJB CORBA objects in the server. CORBAObjectPermission takes two parameters:

- A CORBA object name.
In the Application Server, only the name “*” is supported, that is, it is not possible to specify a specific CORBA object name.
- A comma-separated list of method names.
In the Application Server, only “*” is supported, that is, it is not possible to specify a specific method name.

The general form of a CORBAObjectPermission grant block is:

```
grant principal principal-class-name "principal-name" {
    permission com.sun.enterprise.security.CORBAObjectPermission "*", "*";
}
```

where the *principal-class-name* is either:

- `com.sun.enterprise.deployment.PrincipalImpl` (for a single principal)
- `com.sun.enterprise.deployment.Group` (for a named group of principals)

3 Configure message protection:

Integrity and confidentiality of IIOP messages used in requests and replies during CORBA invocations can be protected by using SSL. By default, the server supports both plain IIOP and IIOP-over-SSL invocations.

- a. **To force clients to use only SSL for IIOP invocations, remove all non-SSL `iiop-listener` elements in the `iiop-service` element in `domain.xml`.**

This ensures that the server will not service plain IIOP invocations. By default, application clients use plain IIOP for making requests if the server supports plain IIOP.

- b. **To force the client to use SSL, a change is needed in the application client configuration file `sun-acc.xml` (which is also located in the domain's config directory).**

Specifically, the property `ssl` with value `required` should be added inside the `<client-container>` element, as follows:

```
<client-container>
  <property name="ssl" value="required"/>
  <target-server ... />
</client-container>
```

Configuring Security for Virtual Servers

▼ To configure single sign-on (SSO)

Single sign-on enables multiple applications to share user sign-on information, rather than requiring each application to have separate user sign-on. Applications using single sign-on authenticate the user one time, and the authentication information is propagated to all other involved applications.

Single sign-on applies to Web applications configured for the same realm and virtual server.

Note – Single sign-on uses an HTTP cookie to transmit a token that associates each request with the saved user identity, so it can be used only when the browser client supports cookies.

Single sign-on operates according to the following rules:

- When a user accesses a protected resource in a Web application, the server requires the user to authenticate himself or herself, using the method defined for that Web application.
- Once authenticated, the Application Server uses the roles associated with the user for authorization decisions across all Web applications on the virtual server, without challenging the user to authenticate to each application individually.
- When the user logs out of one Web application (explicitly, or because of session expiration), the user's sessions in all Web applications become invalid. Thereafter, the user is required to log in to access a protected resource in any application.

- 1 In the Admin Console tree component, expand the Configuration node.**
- 2 Expand the HTTP Service node.**
- 3 Expand the Virtual Servers node, and select the virtual server to be configured for single sign-on support.**
- 4 Click Add Property.**
A blank property entry is added to the bottom of the list.
- 5 Enter `sso-enable` in the Name field.**
- 6 Enter `false` in the Value field to disable, enter `true` to enable SSO.**
SSO is enabled by default.
- 7 Add or change any other single sign-on properties by clicking Add Property and configuring any applicable SSO properties.**

Valid SSO property names for virtual servers are discussed in the following table.

Property Name	Description	Values
<code>sso-max-inactive-seconds</code>	Number of seconds after which a user's single sign-on record becomes eligible for purging, if no client activity is received. Access to any of the applications on the virtual server keeps the single sign-on record active.	Default is 300 seconds (5 minutes). A higher value provides longer persistence for users, but consumes more memory on the server.
<code>sso-reap-interval-seconds</code>	Interval (in seconds) between purges of expired single sign-on records.	Default is 60.

- 8 Click Save.
- 9 Restart the Application Server if Restart Required displays in the console.

Configuring Connector Connection Pools

- [“About Connector Connection Pools” on page 123](#)
- [“About Security Maps” on page 123](#)

About Connector Connection Pools

A *connector module* (also called a resource adapter) enables J2EE applications to interact with enterprise information systems (EIS). A *connector resource* provides an application with a connection to an EIS. A *connector connection pool* is a group of reusable connections for a particular EIS.

Security maps enables the creation of a mapping between Java EE users and groups and EIS users and groups. Use the Admin Console to create, update, list, and delete security maps for connector connection pools.

Note – In this context, users are referred to as principals. The enterprise information system (EIS) is any system that holds the information. It can be a mainframe, a messaging system, a database system, or an application.

About Security Maps

Use security maps to map the caller identity of the application (principal or user group) to a suitable EIS principal in container-managed transaction-based scenarios. When an application

principal initiates a request to an EIS, the application server first checks for an exact principal using the security map defined for the connector connection pool to determine the mapped back end EIS principal. If there is no exact match, then the application server uses the wild card character specification, if any, to determine the mapped back end EIS principal. Security maps are used when an application user needs to execute EIS operations that require to be executed as a specific identity in the EIS.

Use the following procedures in the Admin Console to manage security maps:

- [“To create a security map” on page 124](#)
- [“To edit a security map” on page 125](#)
- [“To delete a security map” on page 126](#)

▼ **To create a security map**

A security map for a connector connection pool maps application users and groups (principals) to EIS principals. Use a security map when an application user needs to execute EIS operations that require a specific identity in the EIS.

- 1 Expand the Resources node**
- 2 Expand the Connectors node.**
- 3 Select the Connector Connection Pools node.**
- 4 Select a Connector Connection Pool by selecting its name from the list of current pools or create a new connector connection pool by selecting New from the list of current pools.**
- 5 Select the Security Maps page.**
- 6 Click New to create a new Security Map.**
- 7 On the Create Security Map page, enter the following properties.**
 - **Name** – Enter a name to be used to reference this particular security map.
 - **User Groups** – The caller identity of the application to be mapped to a suitable EIS principal. Enter a comma-separated list of application-specific user groups, or enter the wild card asterisk (*) to indicate all users or all user groups. Specify either the Principals or User Groups options, but not both.
 - **Principals** – The caller identity of the application to be mapped to a suitable EIS principal. Enter a comma-separated list of application-specific principals, or enter the wild card asterisk (*) to indicate all principals. Specify either the Principals or User Groups options, but not both.
- 8 In the Backend Principal section, enter the following properties.**

- **Username** – Enter the EIS user name. The enterprise information system (EIS) is any system that holds the information. It can be a mainframe, a messaging system, a database system, or an application.
- **Password** – Enter the password for the EIS user.

9 Click OK to create the security map or Cancel to quit without saving.

More Information Equivalent asadmin command

`create-connector-security-map`

For more information, see [create-connector-security-map\(1\)](#) man page.

▼ To edit a security map

- 1 Expand the Resources node.
- 2 Expand the Connectors node.
- 3 Select the Connector Connection Pools node.
- 4 Select a Connector Connection Pool by selecting its name from the list of current pools.
- 5 Select the Security Maps page.
- 6 On the Security Maps page, select a security map from the list of current security maps.
- 7 On the Edit Security Map page, modify the following properties where needed.
 - **User Groups** – The caller identity of the application to be mapped to a suitable EIS principal. Enter a comma-separated list of application-specific user groups, or enter the wild card asterisk (*) to indicate all users or all user groups. Specify either the Principals or User Groups options, but not both.
 - **Principals** – The caller identity of the application to be mapped to a suitable EIS principal. Enter a comma-separated list of application-specific principals, or enter the wild card asterisk (*) to indicate all principals. Specify either the Principals or User Groups options, but not both.
- 8 In the Backend Principal section, enter the following properties.
 - **Username** – Enter the EIS user name. The enterprise information system (EIS) is any system that holds the information. It can be a mainframe, a messaging system, a database system, or an application.
 - **Password** – Enter the password for the EIS user.

- 9 Click **Save** to save the changes to the security map.

More Information Helpful `asadmin` commands

`list-connector-security-maps` and `update-connector-security-map`

- See Also**
- [“About Connector Connection Pools” on page 123](#)
 - [“About Security Maps” on page 123](#)
 - [“To create a security map” on page 124](#)
 - [“To delete a security map” on page 126](#)

▼ **To delete a security map**

- 1 Expand the **Resources** node.
- 2 Expand the **Connectors** node.
- 3 Select the **Connector Connection Pools** node.
- 4 Select a **Connector Connection Pool** by selecting its name from the list of current pools.
- 5 Select the **Security Maps** page.
- 6 On the **Security Maps** page, click the checkbox to the left of the name of the security map to be deleted.
- 7 Click **Delete**.

More Information Equivalent `asadmin` command

`delete-connector-security-map`

- See Also**
- [“About Connector Connection Pools” on page 123](#)
 - [“About Security Maps” on page 123](#)
 - [“To create a security map” on page 124](#)
 - [“To edit a security map” on page 125](#)

Working with Certificates and SSL

- [“About Certificate Files” on page 127](#)
- [“To change the location of certificate files” on page 127](#)
- [“Using Java Secure Socket Extension \(JSSE\) Tools” on page 128](#)

About Certificate Files

Installation of the Application Server generates a digital certificate in JSSE (Java Secure Socket Extension) format suitable for internal testing. By default, the Application Server stores its certificate information in two files in the *domain-dir/config* directory:

- **Keystore file**, `keystore.jks`, contains the Application Server’s certificate, including its private key. The keystore file is protected with a password, initially `changeit`. Change the password using `keytool`. For more information about `keytool`, read [“Using the keytool Utility” on page 128](#).

Each keystore entry has a unique alias. After installation, the Application Server keystore has a single entry with alias `slas`.

- **Truststore file**, `cacerts.jks`, contains the Application Server’s trusted certificates, including public keys for other entities. For a trusted certificate, the server has confirmed that the public key in the certificate belongs to the certificate’s owner. Trusted certificates generally include those of certification authorities (CAs).

In the Platform Edition, on the server side, the Application Server uses the JSSE format, which uses `keytool` to manage certificates and key stores. In the Enterprise Edition, on the server side, the Application Server uses NSS, which uses `certutil` to manage the NSS database which stores private keys and certificates. In both editions, the client side (appclient or stand-alone), uses the JSSE format.

By default, the Application Server is configured with a keystore and truststore that will work with the example applications and for development purposes. For production purposes, you may wish to change the certificate alias, add other certificates to the truststore, or change the name and/or location of the keystore and truststore files.

See also [“To change the location of certificate files” on page 127](#).

▼ To change the location of certificate files

The keystore and truststore files provided for development are stored in the *domain-dir/config* directory.

- 1 In the Admin Console tree, select the Application Server node.
- 2 Select JVM Settings.

- 3 Click the JVM Options tab.
- 4 On the JVM Options page, add or modify the following values in the Value field to reflect the new location of the certificate files:

```
-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/path/ks-name  
-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/path/ts-name
```

where *ks-name* is the keystore file name and *ts-name* is the trust store file name.
- 5 Click Save.
- 6 Restart the Application Server if Restart Required displays in the console.

- See Also**
- [“About Certificate Files” on page 127](#)
 - [“Using Java Secure Socket Extension \(JSSE\) Tools” on page 128](#)

Using Java Secure Socket Extension (JSSE) Tools

Use `keytool` to set up and work with JSSE (Java Secure Socket Extension) digital certificates. In the Platform Edition, the Application Server uses the JSSE format on the server side to manage certificates and key stores. In both the Platform Edition and Enterprise Edition, the client side (appclient or stand-alone) uses the JSSE format.

The J2SE SDK ships with `keytool`, which enables the administrator to administer public/private key pairs and associated certificates. It also enables users to cache the public keys (in the form of certificates) of their communicating peers.

To run `keytool`, the shell environment must be configured so that the `J2SE/bin` directory is in the path, or the full path to the tool must be present on the command line. For more information on `keytool`, see the `keytool` documentation at <http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html>.

Using the keytool Utility

The following examples demonstrate usage related to certificate handling using JSSE tools:

- Create a self-signed certificate in a keystore of type JKS using an RSA key algorithm. RSA is public-key encryption technology developed by RSA Data Security, Inc. The acronym stands for Rivest, Shamir, and Adelman, the inventors of the technology.

```
keytool -genkey -noprompt -trustcacerts -keyalg RSA -alias ${cert.alias}  
-dnname ${dn.name} -keypass ${key.pass} -keystore ${keystore.file}  
-storepass ${keystore.pass}
```

Another example of creating a certificate is shown in [“To generate a certificate using the keytool utility” on page 130](#).

- Create a self-signed certificate in a keystore of type JKS using the default key algorithm.

```
keytool -genkey -noprompt -trustcacerts -alias ${cert.alias} -dname
${dn.name} -keypass ${key.pass} -keystore ${keystore.file} -storepass
${keystore.pass}
```

An example of signing a certificate is shown in [“To sign a digital certificate using the keytool utility” on page 131](#)

- Display available certificates from a keystore of type JKS.

```
keytool -list -v -keystore ${keystore.file} -storepass ${keystore.pass}
```

- Display certificate information from a keystore of type JKS.

```
keytool -list -v -alias ${cert.alias} -keystore ${keystore.file}
-storepass ${keystore.pass}
```

- Import an RFC/text-formatted certificate into a JKS store. Certificates are often stored using the printable encoding format defined by the Internet RFC (Request for Comments) 1421 standard instead of their binary encoding. This certificate format, also known as *Base 64 encoding*, facilitates exporting certificates to other applications by email or through some other mechanism.

```
keytool -import -noprompt -trustcacerts -alias ${cert.alias} -file
${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
```

- Export a certificate from a keystore of type JKS in PKCS7 format. The reply format defined by the Public Key Cryptography Standards #7, Cryptographic Message Syntax Standard, includes the supporting certificate chain in addition to the issued certificate.

```
keytool -export -noprompt -alias ${cert.alias} -file ${cert.file}
-keystore ${keystore.file} -storepass ${keystore.pass}
```

- Export a certificate from a keystore of type JKS in RFC/text format.

```
keytool -export -noprompt -rfc -alias ${cert.alias} -file
${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
```

- Delete a certificate from a keystore of type JKS.

```
keytool -delete -noprompt -alias ${cert.alias} -keystore ${keystore.file}
-storepass ${keystore.pass}
```

Another example of deleting a certificate from a keystore is shown in [“Deleting a Certificate Using the keytool Utility” on page 131](#)

▼ To generate a certificate using the keytool utility

Use `keytool` to generate, import, and export certificates. By default, `keytool` creates a keystore file in the directory where it is run.

1 Change to the directory where the certificate is to be run.

Always generate the certificate in the directory containing the keystore and truststore files, by default *domain-dir/config*. For information on changing the location of these files, see [“To change the location of certificate files” on page 127](#).

2 Enter the following `keytool` command to generate the certificate in the keystore file, `keystore.jks`:

```
keytool -genkey -alias keyAlias -keyalg RSA -keypass changeit -storepass changeit -keystore keystore.jks
```

Use any unique name as your *keyAlias*. If you have changed the keystore or private key password from their default, then substitute the new password for `changeit` in the above command.

A prompt appears that asks for your name, organization, and other information that `keytool` uses to generate the certificate.

3 Enter the following `keytool` command to export the generated certificate to the file `server.cer` (or `client.cer` if you prefer):

```
keytool -export -alias keyAlias -storepass changeit  
-file server.cer  
-keystore keystore.jks
```

4 If a certificate signed by a certificate authority is required, see [“To sign a digital certificate using the keytool utility” on page 131](#).

5 To create the truststore file `cacerts.jks` and add the certificate to the truststore, enter the following `keytool` command:

```
keytool -import -v -trustcacerts -alias keyAlias -file server.cer -keystore cacerts.jks -keypass changeit
```

If you have changed the keystore or private key password from their default, then substitute the new password for `changeit` in the above command.

The tool displays information about the certificate and prompts whether you want to trust the certificate.

6 Type `yes`, then press `Enter`.

Then `keytool` displays something like this:

```
Certificate was added to keystore [Saving cacerts.jks]
```

7 Restart the Application Server.

▼ To sign a digital certificate using the keytool utility

After creating a digital certificate, the owner must sign it to prevent forgery. E-commerce sites, or those for which authentication of identity is important can purchase a certificate from a well-known Certificate Authority (CA). If authentication is not a concern, for example if private secure communications is all that is required, save the time and expense involved in obtaining a CA certificate and use a self-signed certificate.

1 Follow the instructions on the CA's Web site for generating certificate key pairs.

2 Download the generated certificate key pair.

Save the certificate in the directory containing the keystore and truststore files, by default *domain-dir/config* directory. See “[To change the location of certificate files](#)” on page 127.

3 In your shell, change to the directory containing the certificate.

4 Use keytool to import the certificate into the local keystore and, if necessary, the local truststore.

```
keytool -import -v -trustcacerts -alias keyAlias -file server.cer -keystore cacerts.jks -keypass changeit -storepass
```

If the keystore or private key password is not the default password, then substitute the new password for *changeit* in the above command.

5 Restart the Application Server.

Deleting a Certificate Using the keytool Utility

To delete an existing certificate, use the `keytool -delete` command, for example:

```
keytool -delete -alias keyAlias -keystore keystore-name -storepass password
```

For a complete list of possible options for the `-delete` command, refer to the `keytool` documentation at

<http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html>.

Further Information

- The Java 2 Standard Edition discussion on security can be viewed from <http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html>.
- The chapter titled *Security* in the *Java EE 5 Tutorial* (<http://java.sun.com/javaee/5/docs/tutorial/doc/index.html>).
- The *Administration Guide* chapter titled [Chapter 9, “Configuring Message Security.”](#)
- The chapter titled [Chapter 5, “Securing Applications,”](#) in *Sun Java System Application Server Platform Edition 9 Developer's Guide* in the Developer's Guide.

Configuring Message Security

This chapter describes the configuration of message layer security for web services in the Application Server. This chapter contains the following topics:

- “About Application Server Security” on page 89
- “Configuring Security” on page 106

Some of the material in this chapter assumes a basic understanding of security and web services concepts. To learn more about these concepts, explore the resources listed in “[Further Information](#)” on page 131 before beginning this chapter.

About Message Security

- “Overview of Security” on page 89
- “Understanding Message Security in the Application Server” on page 134
- “Securing a Web Service” on page 138
- “Securing the Sample Application” on page 139
- “Configuring the Application Server for Message Security” on page 140

Overview of Message Security

In *message security*, security information is inserted into messages so that it travels through the networking layers and arrives with the message at the message destination(s). Message security differs from transport layer security (which is discussed in the *Security* chapter of the *Java EE 5 Tutorial*) in that message security can be used to decouple message protection from message transport so that messages remain protected after transmission.

Web Services Security: SOAP Message Security (WS-Security) is an international standard for interoperable Web Services Security that was developed in OASIS by a collaboration of all the major providers of web services technology (including Sun Microsystems). WS-Security is a message security mechanism that uses XML Encryption and XML Digital Signature to secure

web services messages sent over SOAP. The WS-Security specification defines the use of various security tokens including X.509 certificates, SAML assertions, and username/password tokens to authenticate and encrypt SOAP web services messages.

The WS-Security specification can be viewed at <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>.

See Also:

- “Understanding Message Security in the Application Server” on page 134
- “Securing a Web Service” on page 138
- “Securing the Sample Application” on page 139
- “Configuring the Application Server for Message Security” on page 140

Understanding Message Security in the Application Server

The Application Server offers integrated support for the WS-Security standard in its web services client and server-side containers. This functionality is integrated such that web services security is enforced by the containers of the Application Server on behalf of applications, and such that it can be applied to protect any web service application without requiring changes to the implementation of the application. The Application Server achieves this effect by providing facilities to bind SOAP layer message security providers and message protection policies to containers and to applications deployed in containers.

Assigning Message Security Responsibilities

In the Application Server, the “[System Administrator](#)” on page 93 and “[Application Deployer](#)” on page 93 roles are expected to take primary responsibility for configuring message security. In some situations, the “[Application Developer](#)” on page 93 may also contribute, although in the typical case either of the other roles may secure an existing application without changing its implementation without involving the developer. The responsibilities of the various roles are defined in the following sections:

- “[System Administrator](#)” on page 93
- “[Application Deployer](#)” on page 93
- “[Application Developer](#)” on page 93

System Administrator

The system administrator is responsible for:

- Configuring message security providers on the Application Server.
- Managing user databases.
- Managing keystore and truststore files.

- Configuring a Java Cryptography Extension (JCE) provider if using encryption
- Installing the samples server. This is only done if the `xms` sample application will be used to demonstrate the use of message layer web services security.

A system administrator uses the Admin Console to manage server security settings and uses a command line tool to manage certificate databases. In Platform Edition, certificates and private keys are stored in key stores and are managed with `keytool`. Standard Edition and Enterprise Edition store certificates and private keys in an NSS database, where they are managed using `certutil`. This document is intended primarily for system administrators. For an overview of message security tasks, see [“Configuring the Application Server for Message Security” on page 140](#).

Application Deployer

The application deployer is responsible for:

- Specifying (at application assembly) any required application-specific message protection policies if such policies have not already been specified by upstream roles (the developer or assembler).
- Modifying Sun-specific deployment descriptors to specify application-specific message protection policies information (message-security-binding elements) to web service endpoint and service references.

These security tasks are discussed in the [Chapter 5, “Securing Applications,” in *Sun Java System Application Server Platform Edition 9 Developer’s Guide*](#) *Securing Applications* chapter of the *Developers’ Guide*.

Application Developer

The application developer can turn on message security, but is not responsible for doing so. Message security can be set up by the System Administrator so that all web services are secured, or by the Application Deployer when the provider or protection policy bound to the application must be different from that bound to the container.

The application developer or assembler is responsible for the following:

- Determining if an application-specific message protection policy is required by the application. If so, ensuring that the required policy is specified at application assembly which may be accomplished by communicating with the Application Deployer.

About Security Tokens and Security Mechanisms

The WS-Security specification provides an extensible mechanism for using security tokens to authenticate and encrypt SOAP web services messages. The SOAP layer message security providers installed with the Application Server may be used to employ username/password and X.509 certificate security tokens to authenticate and encrypt SOAP web services messages.

Additional providers that employ other security tokens including SAML assertions will be installed with subsequent releases of the Application Server.

About Username Tokens

The Application Server uses *Username tokens* in SOAP messages to establish the authentication identity of the message *sender*. The recipient of a message containing a Username token (within embedded password) validates that the message sender is authorized to act as the user (identified in the token) by confirming that the sender knows the secret (the password) of the user.

When using a Username token, a valid user database must be configured on the Application Server. For more information on this topic, read [“Editing a Realm” on page 109](#).

About Digital Signatures

The Application Server uses XML Digital signatures to bind an authentication identity to message *content*. Clients use digital signatures to establish their caller identity, analogous to the way basic authentication or SSL client certificate authentication have been used to do the same thing when transport layer security is being used. Digital signatures are verified by the message receiver to authenticate the source of the message content (which may be different from the sender of the message.)

When using digital signatures, valid keystore and truststore files must be configured on the Application Server. For more information on this topic, read [“About Certificate Files” on page 127](#).

About Encryption

The purpose of encryption is to modify the data such that it can only be understood by its intended audience. This is accomplished by substituting an encrypted element for the original content. When predicated on public key cryptography, encryption can be used to establish the identity of the parties that can read a message.

When using Encryption, you must have an installed JCE provider that supports encryption. For more information on this topic, read [“To configure a JCE Provider” on page 142](#).

About Message Protection Policies

Message protection policies are defined for request message processing and response message processing and are expressed in terms of requirements for source and/or recipient authentication. A source authentication policy represents a requirement that the identity of the entity that sent a message or that defined the content of a message be established in the message such that it can be authenticated by the message receiver. A recipient authentication policy represents a requirement that the message be sent such that the identity of the entities that can

receive the message can be established by the message sender. The providers apply specific message security mechanisms to cause the message protection policies to be realized in the context of SOAP web services messages.

Request and response message protection policies are defined when a provider is configured into a container. Application-specific message protection policies (at the granularity of the web service port or operation) may also be configured within the Sun-specific deployment descriptors of the application or application client. In any case, where message protection policies are defined, the request and response message protection policies of the client must match (be equivalent to) the request and response message protection policies of the server. For more information on defining application-specific message protection policies, refer to the *Securing Applications* chapter of the *Developers' Guide*.

Glossary of Message Security Terminology

The terminology used in this document is described below. The concepts are also discussed in [“Configuring the Application Server for Message Security” on page 140](#).

- Authentication Layer

The *authentication layer* is the message layer on which authentication processing must be performed. The Application Server enforces web services message security at the SOAP layer.

- Authentication Provider

In this release of the Application Server, the Application Server invokes *authentication providers* to process SOAP message layer security.

- A *client-side provider* establishes (by signature or username/password) the source identity of request messages and/or protects (by encryption) request messages such that they can only be viewed by their intended recipients. A client-side provider also establishes its container as an authorized recipient of a received response (by successfully decrypting it) and validates passwords or signatures in the response to authenticate the source identity associated with the response. Client-side providers configured in the Application Server can be used to protect the request messages sent and the response messages received by server-side components (servlets and EJB components) acting as clients of other services.
- A *server-side provider* establishes its container as an authorized recipient of a received request (by successfully decrypting it) and validates passwords or signatures in the request to authenticate the source identity associated with the request. A server-side provider also establishes (by signature or username/password) the source identity of response messages and/or protects (by encryption) response messages such that they can only be viewed by their intended recipients. *Server-side providers* are only invoked by server-side containers.

- Default Server Provider

The *default server provider* is used to identify the server provider to be invoked for any application for which a specific server provider has not been bound. The *default server provider* is sometimes referred to as the *default provider*.

- Default Client Provider

The *default client provider* is used to identify the client provider to be invoked for any application for which a specific client provider has not been bound.

- Request Policy

The *request policy* defines the authentication policy requirements associated with request processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.

- Response Policy

The *response policy* defines the authentication policy requirements associated with response processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.

See Also:

- [“Securing a Web Service” on page 138](#)
- [“Securing the Sample Application” on page 139](#)
- [“Configuring the Application Server for Message Security” on page 140](#)

Securing a Web Service

Web services deployed on the Application Server are secured by binding SOAP layer message security providers and message protection policies to the containers in which the applications are deployed or to web service endpoints served by the applications. SOAP layer message security functionality is configured in the client-side containers of the Application Server by binding SOAP layer message security providers and message protection policies to the client containers or to the portable service references declared by client applications.

When the Application Server is installed, SOAP layer message security providers are configured in the client and server-side containers of the Application Server, where they are available for binding for use by the containers, or by individual applications or clients deployed in the containers. During installation, the providers are configured with a simple message protection policy that, if bound to a container, or to an application or client in a container, would cause the source of the content in all request and response messages to be authenticated by XML digital signature.

The administrative interfaces of the Application Server can be employed to bind the existing providers for use by the server-side containers of the Application Server, to modify the message protection policies enforced by the providers, or to create new provider configurations with

alternative message protection policies. These operations are defined in [“Configuring Security” on page 106](#). Analogous administrative operations can be performed on the SOAP message layer security configuration of the application client container as defined in [“To enable message security for application clients” on page 151](#).

By default, message layer security is disabled on the Application Server. To configure message layer security for the Application Server follow the steps outlined in [“Configuring the Application Server for Message Security” on page 140](#). If you want to cause web services security to be used to protect all web services applications deployed on the Application Server, follow the steps in [“To enable providers for message security” on page 144](#).

Once you have completed the above steps (which may include restarting the Application Server), web services security will be applied to all web services applications deployed on the Application Server.

Configuring Application-Specific Web Services Security

Application-specific web services security functionality is configured (at application assembly) by defining message-security-binding elements in the Sun-specific deployment descriptors of the application. These message-security-binding elements are used to associate a specific provider or message protection policy with a web services endpoint or service reference, and may be qualified so that they apply to a specific port or method of the corresponding endpoint or referenced service.

For more information on defining application specific message protection policies, refer to the *Securing Applications* chapter of the *Developers’ Guide*. There is a link to this chapter in [“Further Information” on page 131](#).

See Also:

- [“To enable providers for message security” on page 144](#)
- [“To enable message security for application clients” on page 151](#)
- [“To configure a message security provider” on page 145](#)

Securing the Sample Application

The Application Server ships with a sample application named `xms`. The `xms` application features a simple web service that is implemented by both a J2EE EJB endpoint and a Java Servlet endpoint. Both endpoints share the same service endpoint interface. The service endpoint interface defines a single operation, `sayHello`, which takes a string argument, and returns a `String` composed by pre-pending `Hello` to the invocation argument.

The `xms` sample application is provided to demonstrate the use of the Application Server’s WS-Security functionality to secure an existing web services application. The instructions which accompany the sample describe how to enable the WS-Security functionality of the

Application Server such that it is used to secure the xms application. The sample also demonstrates the binding of WS-Security functionality directly to the application (as described in [“Configuring Application-Specific Web Services Security” on page 139](#) application.

The xms sample application is installed in the directory:
`install-dir/samples/webservices/security/ejb/apps/xms/`.

For information on compiling, packaging, and running the xms sample application, refer to the *Securing Applications* chapter of the *Developers’ Guide*.

Configuring the Application Server for Message Security

- [“Actions of Request and Response Policy Configurations” on page 140](#)
- [“To configure other security facilities” on page 141](#)
- [“To configure a JCE Provider” on page 142](#)

Actions of Request and Response Policy Configurations

The following table shows message protection policy configurations and the resulting message security operations performed by the WS-Security SOAP message security providers for that configuration.

TABLE 9–1 Message protection policy to WS-Security SOAP message security operation mapping

Message Protection Policy	Resulting WS-Security SOAP message protection operations
auth-source="sender"	The message contains a wsse:Security header that contains a wsse:UsernameToken (with password).
auth-source="content"	The content of the SOAP message Body is signed. The message contains a wsse:Security header that contains the message Body signature represented as a ds:Signature.
auth-source="sender" auth-recipient="before-content" OR auth-recipient="after-content"	The content of the SOAP message Body is encrypted and replaced with the resulting xenc:EncryptedData. The message contains a wsse:Security header that contains a wsse:UsernameToken (with password) and an xenc:EncryptedKey. The xenc:EncryptedKey contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.

TABLE 9-1 Message protection policy to WS-Security SOAP message security operation mapping
(Continued)

Message Protection Policy	Resulting WS-Security SOAP message protection operations
auth-source="content" auth-recipient="before-content"	The content of the SOAP message Body is encrypted and replaced with the resulting <code>xenc:EncryptedData</code> . The <code>xenc:EncryptedData</code> is signed. The message contains a <code>wsse:Security</code> header that contains an <code>xenc:EncryptedKey</code> and a <code>ds:Signature</code> . The <code>xenc:EncryptedKey</code> contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-source="content" auth-recipient="after-content"	The content of the SOAP message Body is signed, then encrypted, and then replaced with the resulting <code>xenc:EncryptedData</code> . The message contains a <code>wsse:Security</code> header that contains an <code>xenc:EncryptedKey</code> and a <code>ds:Signature</code> . The <code>xenc:EncryptedKey</code> contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-recipient="before-content" OR auth-recipient="after-content"	The content of the SOAP message Body is encrypted and replaced with the resulting <code>xenc:EncryptedData</code> . The message contains a <code>wsse:Security</code> header that contains an <code>xenc:EncryptedKey</code> . The <code>xenc:EncryptedKey</code> contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
No policy specified.	No security operations are performed by the modules.

▼ To configure other security facilities

The Application Server implements message security using message security providers integrated in its SOAP processing layer. The message security providers depend on other security facilities of Application Server.

- 1 If using a version of the Java SDK prior to version 1.5.0, and using encryption technology, configure a JCE provider.**
Configuring a JCE provider is discussed in [“To configure a JCE Provider” on page 142](#).
- 2 If using a username token, configure a user database, if necessary. When using a username/password token, an appropriate realm must be configured and an appropriate user database must be configured for the realm.**
Configuring a user database is discussed in [“Editing a Realm” on page 109](#).
- 3 Manage certificates and private keys, if necessary.**
Managing certificates and private keys is discussed in [“About Certificate Files” on page 127](#).

Next Steps Once the facilities of the Application Server are configured for use by message security providers, then the providers installed with the Application Server may be enabled as described in [“To enable providers for message security” on page 144](#).

▼ To configure a JCE Provider

The Java Cryptography Extension (JCE) provider included with J2SE 1.4.x does not support RSA encryption. Because the XML Encryption defined by WS-Security is typically based on RSA encryption, in order to use WS-Security to encrypt SOAP messages you must download and install a JCE provider that supports RSA encryption.

Note – RSA is public-key encryption technology developed by RSA Data Security, Inc. The acronym stands for Rivest, Shamir, and Adelman, the inventors of the technology.

If you are running the Application Server on version 1.5 of the Java SDK, the JCE provider is already configured properly. If you are running the Application Server on version 1.4.x of the Java SDK, you can add a JCE provider statically as part of your JDK environment, as follows.

1 Download and install a JCE provider JAR (Java ARchive) file.

The following URL provides a list of JCE providers that support RSA encryption:
http://java.sun.com/products/jce/jce14_providers.html.

2 Copy the JCE provider JAR file to *java-home/jre/lib/ext/*.

3 Stop the Application Server.

If the Application Server is not stopped and then restarted later in this process, the JCE provider will not be recognized by the Application Server.

4 Edit the *java-home/jre/lib/security/java.security* properties file in any text editor. Add the JCE provider you've just downloaded to this file.

The *java.security* file contains detailed instructions for adding this provider. Basically, you need to add a line of the following format in a location with similar properties:

```
security.provider.n=provider-class-name
```

In this example, *n* is the order of preference to be used by the Application Server when evaluating security providers. Set *n* to 2 for the JCE provider you've just added.

For example, if you've downloaded The Legion of the Bouncy Castle JCE provider, you would add this line.

```
security.provider.2=org.bouncycastle.jce.provider.  
    BouncyCastleProvider
```

Make sure that the Sun security provider remains at the highest preference, with a value of 1.

```
security.provider.1=sun.security.provider.Sun
```

Adjust the levels of the other security providers downward so that there is only one security provider at each level.

The following is an example of a `java.security` file that provides the necessary JCE provider and keeps the existing providers in the correct locations.

```
security.provider.1=sun.security.provider.Sun
security.provider.2=org.bouncycastle.jce.provider.
    BouncyCastleProvider
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.rsa.jca.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
```

- 5 **Save and close the file.**
- 6 **Restart the Application Server.**

Admin Console Tasks for Message Security

Most of the steps for setting up the Application Server for using message security can be accomplished using the Admin Console, the `asadmin` command-line tool, or by manually editing system files. In general, editing system files is discouraged due to the possibility of making unintended changes that prevent the Application Server from running properly, therefore, where possible, steps for configuring the Application Server using the Admin Console are shown first, with the `asadmin` tool command shown after. Steps for manually editing system files are shown only when there is no Admin Console or `asadmin` equivalent.

Support for message layer security is integrated into the Application Server and its client containers in the form of (pluggable) authentication modules. By default, message layer security is disabled on the Application Server. The following sections provide the details for enabling, creating, editing, and deleting message security configurations and providers.

- [“To enable providers for message security” on page 144](#)
- [“To configure a message security provider” on page 145](#)
- [“Creating a Message Security Provider” on page 148](#)
- [“To delete a message security configuration” on page 151](#)
- [“To delete a message security provider” on page 151](#)
- [“To enable message security for application clients” on page 151](#)

In most cases, it will be necessary to restart the Application Server after performing the administrative operations listed above. This is especially the case if you want the effects of the administrative change to be applied to applications that were already deployed on the Application Server at the time the operation was performed.

▼ To enable providers for message security

To enable message security for web services endpoints deployed in the Application Server, you must specify a provider to be used by default on the server side. If you enable a default provider for message security, you also need to enable providers to be used by clients of the web services deployed in the Application Server. Information for enabling the providers used by clients is discussed in [“To enable message security for application clients” on page 151](#).

To enable message security for web service invocations originating from deployed endpoints, you must specify a default client provider. If you enabled a default client provider for the Application Server, you must ensure that any services invoked from endpoints deployed in the Application Server are compatibly configured for message layer security.

- 1 In the Admin Console tree component, expand the Configuration node.
- 2 Expand the Security node.
- 3 Expand the Message Security node.
- 4 Select the SOAP node.
- 5 Select the Message Security tab.
- 6 On the Edit Message Security Configuration page, specify a provider to be used on the server side and a provider to be used on the client side for all applications for which a specific provider has not been bound.

This is accomplished by modifying the following optional properties:

- **Default Provider** – The identity of the server provider to be invoked for any application for which a specific server provider has not been bound.

By default, no provider configuration is selected for the Application Server. To identify a server-side provider, select `ServerProvider`. Selecting the null option means that no message security provider will be invoked (by default) on the server side.

You would generally select `ServerProvider` for this field.

- **Default Client Provider** – The identity of the client provider to be invoked for any application for which a specific client provider has not been bound.

By default, no provider configuration is selected for the Application Server. To identify a client-side provider, select `ClientProvider`. Selecting the null option means that no message security provider will be invoked (by default) on the client side.

You would generally select null for this field. You would select `ClientProvider` if you wanted to enable a default provider and message protection policy to apply to the web services invocations originating from web services endpoints deployed on the Application Server.

- 7 Click Save.
- 8 If you enabled a client or server provider and you want to modify the message protection policies of the enabled providers, refer to [“To configure a message security provider” on page 145](#) for information on modifying the message security providers enabled in this step.

More Information Equivalent asadmin commands

- To specify the default server provider:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_provider=ServerProvider
```

- To specify the default client provider:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_client_provider=ClientProvider
```

- See Also**
- [“To configure a message security provider” on page 145](#)
 - [“Creating a Message Security Provider” on page 148](#)
 - [“To delete a message security configuration” on page 151](#)
 - [“To enable message security for application clients” on page 151](#)

▼ To configure a message security provider

Typically, a provider would be reconfigured to modify its message protection policies, although the provider type, implementation class, and provider-specific configuration properties may also be modified.

- 1 In the Admin Console tree component, expand the Configuration node.
- 2 Expand the Security node.
- 3 Expand the Message Security node.
- 4 Select the SOAP node.
- 5 Select the Providers tab.
- 6 Select the message security provider to edit.
ClientProvider and ServerProvider ship with the Application Server.

7 In the Provider Configuration section of the Edit Provider Configuration page, the following properties are available for modification:

- **Provider Type** – Select `client`, `server`, or `client-server` to establish whether the provider is to be used as a client authentication provider, a server authentication provider, or both (a client-server provider).
- **Class Name** – Enter the Java implementation class of the provider. Client authentication providers must implement the `com.sun.enterprise.security.jauth.ClientAuthModule` interface. Server-side providers must implement the `com.sun.enterprise.security.jauth.ServerAuthModule` interface. A provider may implement both interfaces, but it must implement the interface corresponding to its provider type.

8 In the Request Policy section of the Edit Provider Configuration page, enter the following optional values, if needed.

These properties are optional, but if not specified, no authentication is applied to request messages.

The *request policy* defines the authentication policy requirements associated with request processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.

- **Authentication Source**– Select `content` or `null` (the blank option) to define a requirement for message-layer content authentication (for example, digital signature) to be applied to response messages. When `null` is specified, source authentication of the response is not required. sender is not supported.
- **Authentication Recipient**– Select `beforeContent` or `afterContent` to define a requirement for message-layer authentication of the receiver of the request message to its sender (by XML encryption). When the value is not specified it defaults to `afterContent`.

For a description of the actions performed by the SOAP message security providers as a result of the following message protection policies see [“Actions of Request and Response Policy Configurations” on page 140](#).

9 In the Response Policy section of the Create a Provider Configuration page, enter the following optional properties, if needed.

These properties are optional, but if not specified, no authentication is applied to response messages.

The *response policy* defines the authentication policy requirements associated with response processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.

- **Authentication Source** – Select content or null (the blank option) to define a requirement for message-layer content authentication (for example, digital signature) to be applied to response messages. When null is specified, source authentication of the response is not required.
- **Authentication Recipient** – Select beforeContent or afterContent to define a requirement for message-layer authentication of the receiver of the response message to its sender (by XML encryption). When the value is not specified it defaults to afterContent.

For a description of the actions performed by the SOAP message security providers as a result of the following message protection policies see [“Actions of Request and Response Policy Configurations” on page 140](#).

10 Add additional properties by clicking the Add Property button.

The provider that is shipped with the Application Server supports the property listed below. If other providers are used, refer to their documentation for more information on properties and valid values.

- `security.config` – The directory and file name of an XML file that contains the server configuration information. For example, `domain-dir/config/wss-server-config-2.0.xml`.

11 Click Save.

More Information Equivalent asadmin commands

To set the response policy, replace the word request in the following commands with response.

- Add a request policy to the client and set the authentication source:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_source=
sender | content
```

- Add a request policy to the server and set the authentication source:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_source=
sender | content
```

- Add a request policy to the client and set the authentication recipient:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_recipient=
before-content | after-content
```

- Add a request policy to the server and set the authentication recipient:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_recipient=
before-content | after-content
```

▼ Creating a Message Security Provider

To configure an existing provider, follow the steps in [“To configure a message security provider” on page 145](#).

- 1 In the Admin Console tree component, expand the Configuration node.
- 2 Expand the Security node.
- 3 Expand the Message Security node.
- 4 Select the SOAP node.
- 5 Select the Providers tab.
- 6 On the Provider Configuration page, click New.
- 7 In the Provider Configuration section of the New Provider Configuration page, enter the following:
 - **Default Provider** – Check the box beside this field to make the new message security provider the provider to be invoked for any application for which a specific provider has not been bound. Whether the provider becomes the default client provider, the default server provider, or both will be based on the value selected for Provider Type.
 - **Provider Type** – Select client, server, or client-server to establish whether the provider is to be used as a client authentication provider, a server authentication provider, or both (a client-server provider).
 - **Provider ID** - Enter an identifier for this provider configuration. This name will appear in the Current Provider Configurations list.
 - **Class Name** - Enter the Java implementation class of the provider. Client authentication providers must implement the `com.sun.enterprise.security.jauth.ClientAuthModule` interface. Server-side providers must implement the `com.sun.enterprise.security.jauth.ServerAuthModule` interface. A provider may implement both interfaces, but it must implement the interface corresponding to its provider type.

8 In the Request Policy section of the Create a Provider Configuration page, enter the following optional values, if needed.

These properties are optional, but if not specified, no authentication is applied to request messages.

- **Authentication Source** – Select sender, content, or null (the blank option) to define a requirement for message-layer sender authentication (for example, username password), content authentication (for example, digital signature), or no authentication be applied to request messages. When null is specified, source authentication of the request is not required.
- **Authentication Recipient** – Select beforeContent or afterContent to define a requirement for message-layer authentication of the receiver of the request message to its sender (by XML encryption). When the value is not specified it defaults to afterContent.

For a description of the actions performed by the SOAP message security providers as a result of the following message protection policies see [“Actions of Request and Response Policy Configurations” on page 140](#).

9 In the Response Policy section of the Create a Provider Configuration page, enter the following optional properties, if needed.

These properties are optional, but if not specified, no authentication is applied to response messages.

- **Authentication Source** – Select sender, content, or null (the blank option) to define a requirement for message-layer sender authentication (for example, username password) or content authentication (for example, digital signature) to be applied to response messages. When null is specified, source authentication of the response is not required. sender is not supported.
- **Authentication Recipient** – Select beforeContent or afterContent to define a requirement for message-layer authentication of the receiver of the response message to its sender (by XML encryption). When the value is not specified it defaults to afterContent.

For a description of the actions performed by the SOAP message security providers as a result of the following message protection policies see [“Actions of Request and Response Policy Configurations” on page 140](#).

10 Add additional properties by clicking the Add Property button.

The provider that is shipped with the Application Server supports the property listed below. If other providers are used, refer to their documentation for more information on properties and valid values.

- `security.config` – The directory and file name of an XML file that contains the server configuration information. For example,
`domain-dir/config/wss-server-config-2.0.xml`.

11 Click OK to save this configuration, or click Cancel to quit without saving.

More Information Equivalent asadmin command
`create-message-security-provider`

▼ To create an `httpServlet` provider

- 1 In the Admin Console tree component, expand the Configuration node.
- 2 Expand the Security node.
- 3 Expand the Message Security node.
- 4 On the Message Security Configuration page, click New.
- 5 In the Authentication Layer drop-down list, select `HttpServlet`.
- 6 In the Provider Configuration section, specify the following:
 - **Default Provider** – Check the box beside this field to make the new message security provider the provider to be invoked for any application for which a specific provider has not been bound. Whether the provider becomes the default client provider, the default server provider, or both will be based on the value selected for Provider Type.
 - **Provider Type** – Select server to specify that the provider is to be used as a server authentication provider.
 - **Provider ID** - Enter `JSR196AMProvider` as the name of this provider configuration. This name will appear in the Current Provider Configurations list.
 - **Class Name** - Enter the Java implementation class of the provider, which is `com.sun.identity.agents.jsr196.as9.JSR196AMAuthModule`.
- 7 Add additional properties by clicking the Add Property button.
- 8 Click OK to create the provider. The new provider `HttpServlet` is listed in the Message Security Configuration page as well as a new node under the Message Security node.
- 9 Click the `HttpServlet` node under the Message Security node.
- 10 Click the Providers tab and click `HttpServlet` to open the Edit Provider Configuration page.
- 11 In the Request Policy section, click `Authenticate Source` and select content.
- 12 In the Response Policy section, click `Authenticate Source` and select content.
- 13 Click Save to complete the message security provider creation.

More Information Equivalent asadmin command
`create-message-security-provider`

▼ To delete a message security configuration

- 1 In the Admin Console tree component, expand the Configuration node.
- 2 Expand the Security node.
- 3 Select the Message Security node.
- 4 Click in the checkbox to the left of the Message Security Configuration to be deleted.
- 5 Click Delete.

▼ To delete a message security provider

- 1 In the Admin Console tree component, expand the Configuration node.
- 2 Expand the Security node.
- 3 Expand the Message Security node.
- 4 Select the SOAP node.
- 5 Select the Providers page.
- 6 Click in the checkbox to the left of the Provider Configuration to be deleted.
- 7 Click Delete.

More Information Equivalent asadmin command
`delete-message-security-provider`

▼ To enable message security for application clients

The message protection policies of client providers must be configured such that they are equivalent to the message protection policies of the server-side providers they will be interacting with. This is already the case for the providers configured (but not enabled) when the Application Server is installed.

To enable message security for client applications, modify the Application Server specific configuration for the application client container.

- 1 **Stop any client applications that depend on the client container descriptor.**
 - 2 **In a text editor, open the Sun application client container descriptor, located in `domain-dir/config/sun-acc.xml`.**
 - 3 **Add the `default-client-provider` attribute in the `message-security` element to the `sun-acc.xml` file to enable the default client provider in the application client.**
- The rest of the code shown here may differ slightly in your installation. Do not change it.

```
<client-container>
  <target-server name="your-host" address="your-host"
    port="your-port"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
    default-client-provider="ClientProvider">
    <provider-config
      class-name="com.sun.enterprise.security.jauth.ClientAuthModule"
      provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender"/>
      <response-policy/>
      <property name="security.config"
        value="C:/Sun/AppServer/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

The message security provider configured in the client container will also require access to private keys and trusted certificates. This is accomplished by defining appropriate values for the following system properties in the application client startup script.

```
-Djavax.net.ssl.keyStore
```

```
-Djavax.net.ssl.trustStore
```

Setting the Request and Response Policy for the Application Client Configuration

The *request and response policies* define the authentication policy requirements associated with request and response processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.

To achieve message security, the request and response policies must be enabled on both the server and client. When configuring the policies on the client and server, make sure that the client policy matches the server policy for request/response protection at application-level message binding.

To set the request policy for the application client configuration, modify the Application Server specific configuration for the application client container as described in [“To enable message security for application clients” on page 151](#). In the application client configuration file, add the request-policy and response-policy elements as shown to set the request policy.

The rest of the code shown here may differ slightly in your installation. Do not change it.

```
<client-container>
  <target-server name="your-host" address="your-host"
    port="your-port"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
    default-client-provider="ClientProvider">
    <provider-config
      class-name="com.sun.enterprise.security.jauth.ClientAuthModule"
      provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <property name="security.config"
        value="install-dir/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

Valid values for auth-source include sender and content. Valid values for auth-recipient include before-content and after-content. A table describing the results of various combinations of these values can be found in [“Actions of Request and Response Policy Configurations” on page 140](#).

To not specify a request or response policy, leave the element blank, for example:

```
<response-policy/>
```

Further Information

- The Java 2 Standard Edition discussion of security can be viewed from <http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html>.
- The *Java EE 5 Tutorial* chapter titled *Security* can be viewed from [Java EE 5 Tutorial \(http://java.sun.com/javaee/5/docs/tutorial/doc/index.html\)](http://java.sun.com/javaee/5/docs/tutorial/doc/index.html).

- The *Administration Guide* chapter titled “Overview of Security” on page 89 Chapter 8, “Configuring Security.”
- The *Developer’s Guide* chapter, Chapter 5, “Securing Applications,” in *Sun Java System Application Server Platform Edition 9 Developer’s Guide*.
- The Oasis Web Services Security: SOAP Message Security (WS-Security) specification, can be viewed from <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>.
- The OASIS Web Services Security Username Token Profile 1.0, can be found at <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>.
- The OASIS Web Services Security X.509 Certificate Token Profile 1.0, can be found at <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf>.
- The *XML-Signature Syntax and Processing* document can be viewed at <http://www.w3.org/TR/xmlsig-core/>.
- The *XML Encryption Syntax and Processing* document can be viewed at <http://www.w3.org/TR/xmlenc-core/>.

Transactions

Application Server handles transactions based on Java EE specification requirements. It supports some extra features that are exposed and can be changed in the Administration Console. This chapter discusses the following:

- [“About Transactions” on page 155](#)
- [“Configuring Transaction Management” on page 157](#)

About Transactions

The following subsections explain what are transactions and how transaction processing is handled in Java EE.

- [“What Is a Transaction?” on page 155](#)
- [“Transactions in Java EE Technology” on page 156](#)
- [“Workarounds for Specific Databases” on page 156](#)

What Is a Transaction?

A transaction is a series of discreet actions in an application that must all complete successfully or else all the changes in each action are backed out. For example, to transfer funds from a checking account to a savings account is a transaction with the following steps:

1. Check to see if the checking account has enough money to cover the transfer.
2. If there’s enough money in the checking account debit the amount from the checking account.
3. Credit the money to the savings account.
4. Record the transfer to the checking account log.
5. Record the transfer to the savings account log.

If any of these steps fails, all changes from the preceding steps must be backed out, and the checking account and savings account must be in the same state as they were before the transaction started. This event is called a *rollback*. If all the steps complete successfully, the transaction is in a *committed* state. Transactions end in either a commit or a rollback.

Transactions in Java EE Technology

Transaction processing in Java EE technology involves the following five participants:

- Transaction Manager
- Application Server
- Resource Manager
- Resource Adapter
- User Application

Each of these entities contributes to reliable transaction processing by implementing the different APIs and functionality, discussed below:

- The Transaction Manager provides the services and management functions required to support transaction demarcation, transactional resource management, synchronization, and transaction context propagation.
- The Application Server provides the infrastructure required to support the application runtime environment that includes transaction state management.
- The 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 a 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 a JDBC driver.
- A User Application developed to operate in an application server environment may use transactional data sources. The application may use declarative transaction attribute settings for enterprise beans or explicit programmatic transaction demarcation.

Workarounds for Specific Databases

The Application Server provides workarounds for some known issues with the recovery implementations of the following JDBC drivers. These workarounds are used unless explicitly disabled.

- Oracle thin driver - The `XAResource.recover` method repeatedly returns the same set of in-doubt Xids regardless of the input flag. According to the XA specifications, the Transaction Manager initially calls this method with `TMSTARTSCAN` and then with `TMNOFLAGS` repeatedly until no Xids are returned. The `XAResource.commit` method also has some issues.

To disable the Application Server workaround, set the `oracle-xa-recovery-workaround` property value to `false`. For details about how to set a property, see the Admin Console online help.

Note – These workaround do not imply support for any particular JDBC driver.

Configuring Transaction Management

The Application Server handles transactions based on the settings in the Admin Console.

This section explains how to configure transaction settings:

- [“Configuring Application Server to Recover Transactions” on page 157](#)
- [“Setting a Transaction Timeout Value” on page 158](#)
- [“Specifying the Transaction Logs” on page 159](#)
- [“Setting the Keypoint Interval” on page 159](#)

For additional information about transactions, see these sections:

- [“What Is a Transaction?” on page 155](#)
- [“Transactions in Java EE Technology” on page 156](#)

Configuring Application Server to Recover Transactions

Transactions might be incomplete either because the server crashed or a resource manager crashed. It is essential to complete these stranded transactions and recover from the failures. Application Server is designed to recover from these failures and complete the transactions upon server startup.

While performing the recovery, if some of the resources are unreachable the server restart may be delayed as it tries to recover the transactions.

When the transaction spans across servers, the server that started the transaction can contact the other servers to get the outcome of the transactions. If the other servers are unreachable, the transaction uses the Heuristic Decision field to determine the outcome.

Note – For Oracle, the database user for an XA datasource requires necessary permissions to be able to recover any transactions after Application Server has crashed and has been restarted. The Oracle permissions required on various tables for transaction recovery are:

- SELECT permissions on DBA_PENDING_TRANSACTIONS, PENDING_TRANS\$, DBA_2PC_PENDING and DBA_2PC_NEIGHBORS tables
 - EXECUTE permissions on DBMS_SYSTEM table
-

To configure Application Server to recover transactions using the Administration Console, select Configuration>Transaction Service. The Transaction Service page displays the general transaction service settings.

To enable the recovery of incomplete transactions, check the Recover box in the On Restart field. 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 is marked, the server attempts to recover stranded transactions upon server restart. If the involved resources remain unreachable, this may delay server restart. This checkbox is not marked by default. Note that if automatic recovery is not enabled, only 2048 XA transactions are recoverable. Also see the use of the Keypoint Interval as described below.

Modify values of other fields as necessary. 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.

Detailed steps for doing this are provided in the Administration Console Online Help.

Setting a Transaction Timeout Value

By default, the server does not time out a transaction. That is, the server waits indefinitely for a transaction to complete. If you set a timeout value for transactions, if a transaction isn't completed within the configured time, the Application Server rolls back the transaction.

To set transaction timeout value using the Administration Console, select Configuration>Transaction Service. The Transaction Service page displays the general transaction service settings. Enter the number of seconds before the transaction times out, in the Transaction Timeout field. The default value of Transaction Timeout is 0 seconds. This disables transaction timeouts.

Specifying the Transaction Logs

The transaction log records the information about each transaction in order to maintain the data integrity of the resources involved and to recover from failures. Transaction logs are kept in the tx subdirectory of the directory specified by the Transaction Log Location field. These logs are not human readable.

To set the location of the transaction logs using the Administration Console, select Configuration>Transaction Service. The Transaction Service page displays the general transaction service settings. Enter the location of the transaction logs in the Transaction Log Location field.

A tx subdirectory is created and transaction logs are kept under that directory.

Setting the Keypoint Interval

Keypoint operations compress the transaction log file. The keypoint interval is the number of transactions between keypoint operations on the log. Keypoint operations can reduce the size of the transaction log files. A larger number of keypoint intervals (for example, 2048) results in larger transaction log files, but fewer keypoint operations, and potentially better performance. A smaller keypoint interval (for example, 256) results in smaller log files but slightly reduced performance due to the greater frequency of keypoint operations.

To set the keypoint interval using the Administration Console, select Configuration>Transaction Service. The Transaction Service page displays the general transaction service settings. Enter the number of transactions between keypoint operations in the Keypoint Interval field.

Configuring the HTTP Service

The HTTP service is the component of the Application Server that provides facilities for deploying web applications and for making deployed web applications accessible by HTTP clients. These facilities are provided by means of two kinds of related objects, virtual servers and HTTP listeners.

- [“Virtual Servers” on page 161](#)
- [“HTTP Listeners” on page 162](#)

Virtual Servers

A virtual server, sometimes called a virtual host, is an object that allows the same physical server to host multiple Internet domain names. All virtual servers hosted on the same physical server share the Internet Protocol (IP) address of that physical server. A virtual server associates a domain name for a server (such as `www.aaa.com`) with the particular server on which the Application Server is running.

Note – Do not confuse an Internet domain with the administrative domain of the Application Server.

For instance, assume you want to host these domains on your physical server:

```
www.aaa.com  
www.bbb.com  
www.ccc.com
```

Assume also that `www.aaa.com`, `www.bbb.com`, and `www.ccc.com` have web modules `web1`, `web2`, and `web3`, respectively, associated with them.

This means that all of these URLs are handled by your physical server:

```
http://www.aaa.com:8080/web1
http://www.bbb.com:8080/web2
http://www.ccc.com:8080/web3
```

The first URL is mapped to virtual host `www.aaa.com`, the second URL is mapped to virtual host `www.bbb.com`, and the third is mapped to virtual host `www.ccc.com`.

On the other hand, the following URL results in a 404 return code, because `web3` isn't registered with `www.bbb.com`:

```
http://www.bbb.com:8080/web3
```

For this mapping to work, make sure that `www.aaa.com`, `www.bbb.com`, and `www.ccc.com` all resolve to your physical server's IP address. They need to be registered with the DNS server for your network. In addition, on a UNIX system, add these domains to your `/etc/hosts` file (if the setting for hosts in your `/etc/nsswitch.conf` file includes `files`).

When the Application Server is started, it starts the following virtual servers automatically:

- A virtual server named `server`, which hosts all user-defined web modules
- A virtual server named `__asadmin`, which hosts all administration-related web modules (specifically, the Admin Console). This server is restricted; you cannot deploy web modules to this virtual server.

For development, testing, and deployment of web services in a non-production environment, `server` is often the only virtual server required. In a production environment, additional virtual servers provide hosting facilities for users and customers so that each appears to have its own web server, even though there is only one physical server.

HTTP Listeners

Each virtual server provides connections between the server and clients through one or more HTTP listeners. Each HTTP listener is a listen socket that has an IP address, a port number, a server name, and a default virtual server.

HTTP listeners must have a unique combination of port number and IP address. For example, an HTTP listener can listen on all configured IP addresses on a given port for a machine by specifying the IP address `0.0.0.0`. Alternatively, the HTTP listener can specify a unique IP address for each listener, but use the same port.

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 (for example, `1.1.1.1:8081` and `1.1.1.1:8082`), or with different IP addresses and the same port number (for example, `1.1.1.1:8081` and `1.2.3.4:8081`, if your machine was configured to respond to both these addresses).

However, if an HTTP listener uses the 0.0.0.0 IP address, which listens on all IP addresses on a port, you cannot create HTTP listeners for additional IP addresses that listen on the same port for a specific IP address. For example, if an HTTP listener uses 0.0.0.0:8080 (all IP addresses on port 8080), another HTTP listener cannot use 1.2.3.4:8080.

Because the system running the Application Server typically has access to only one IP address, HTTP listeners typically use the 0.0.0.0 IP address and different port numbers, with each port number serving a different purpose. If the system does have access to more than one IP address, each address can serve a different purpose.

By default, when the Application Server starts, it has the following HTTP listeners:

- Two HTTP listeners named `http-listener-1` and `http-listener-2`, associated with the virtual server named `server`. The listener named `http-listener-1` does not have security enabled; `http-listener-2` has security enabled.
- An HTTP listener named `admin-listener`, associated with the virtual server named `__asadmin`. This listener does not have security enabled.

All these listeners use the IP address 0.0.0.0 and the port numbers specified as the HTTP server port numbers during installation of the Application Server. If the Application Server uses the default port number values, `http-listener-1` uses port 8080, `http-listener-2` uses port 8181, and `admin-listener` uses port 4848.

Each HTTP listener has a default virtual server. The default virtual server is the server to which the HTTP listener routes all request URLs whose host component does not match any of the virtual servers that are associated with the HTTP listener (a virtual server is associated with an HTTP listener by listing the HTTP listener in its `http-listeners` attribute).

In addition, specify the number of acceptor threads in the HTTP listener. Acceptor 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. Configure enough acceptor 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. In the Application Server, there is no distinction between acceptor and request processing (worker) threads: each HTTP listener thread is responsible for accepting and processing requests. For this reason, the HTTP listeners in the Application Server's default configuration use 50 acceptor threads.

The HTTP listener's server name is the host name that appears in the URLs the server sends to the client as part of a redirect. This attribute affects URLs the server automatically generates; it does not affect the URLs for directories and files stored in the server. This name is normally the alias name if the server uses an alias. If a client sends a `Host :` header, that host name supersedes the HTTP listener's server name value in redirects.

Specify a redirect port to use a different port number from that specified in the original request. A *redirect* occurs in one of these situations:

- If a client tries to access a resource that no longer exists at the specified URL (that is, the resource has moved to another location), the server redirects the client to the new location (instead of returning a 404), by returning a designated response code and including the new location in the response's Location header.
- If a client tries to access a resource that is protected (for example, SSL) on the regular HTTP port, the server redirects the request to the SSL-enabled port. In this case, the server returns a new URL in the Location response header, in which the original insecure port has been replaced with the SSL-enabled port. The client then connects to this new URL.

Specify also whether security is enabled for an HTTP listener and what kind of security is used (for example, which SSL protocol and which ciphers).

To access a web application deployed on the Application Server, use the URL `http://localhost:8080/` (or `https://localhost:8181/` if it is a secure application), along with the context root specified for the web application. To access the Admin Console, use the URL `http://localhost:4848/` or `http://localhost:4848/asadmin/` (its default context root).

Because a virtual server must specify an existing HTTP listener, and because it cannot specify an HTTP listener that is already being used by another virtual server, create at least one HTTP listener before creating a new virtual server.

Managing Web Services

This chapter describes web services management with Application Server. Admin Console and the `asadmin` tool enable you to deploy, test, and manage web services. You can view, manage, and monitor all web services deployed in a domain just as you see Java EE applications and application components such as EJBs.

You can also:

- Track response times and invocation counts for web services in real time.
- Generate alerts on boundary conditions including response time and throughput failures. For more information, see [Chapter 17, “Configuring Management Rules.”](#)
- View web service invocation content in XML.
- Transform messages at runtime using XSLT.

This chapter contains the following topics:

- [“Overview of Web Services” on page 166](#)
- [“Deploying and Testing Web Services” on page 168](#)
- [“Publishing to Web Services Registries” on page 170](#)
- [“Monitoring Web Services” on page 174](#)
- [“Transforming Messages with XSLT Filters” on page 175](#)
- [“Web Services as JBI Service Providers” on page 176](#)

For information on securing web services, see [Chapter 9, “Configuring Message Security.”](#)

Overview of Web Services

Extensible Markup Language (XML), developed by the World Wide Web Consortium (W3C) is one of the foundations of web services technology. XML enables web services and clients to communicate with each other in a common language. XML is a simple, flexible, text-based markup language in which data are marked using tags enclosed in angle brackets. Such markup allows different systems to easily exchange data with each other.

A web service is an application accessed by clients using XML-based protocols, such as Simple Object Access Protocol (SOAP), sent over internet protocols such as Hypertext Transfer Protocol (HTTP). Clients access a web service application through its interfaces and bindings, defined using XML artifacts such as a web services Definition Language (WSDL) file.

A Document Type Definition (DTD) or XML Schema Definition (XSD) describes the structure of an XML document. It has the tags an XML document can have, the order of those tags, and so on.

Extensible Stylesheet Language Transformation (XSLT) transforms XML documents from one format to another.

Web Services Standards

Simple Object Access Protocol (SOAP) provides a common messaging format for web services. SOAP enables objects not known to one another to exchange messages. SOAP uses an XML-based data encoding format and HTTP to transport messages. SOAP is independent of both the programming language and the operational platform, and it does not require any specific technology at its endpoints.

ebXML (Electronic Business using eXtensible Markup Language) is a set of specifications that enables enterprises to conduct business over the Internet. [OASIS](#) (Organization for the Advancement of Structured Information Standards) controls the ebXML specifications. Similar to a telephone system's yellow pages, an ebXML registry enables providers to register their services and requestors to find services. Once a requestor finds a service, the registry has no more role to play between the requestor and the provider.

Universal Description, Discovery, and Integration (UDDI) provides a standard way to register, de-register, and look up web services.

Web Services Description Language (WSDL) defines a standard way to specify the details of a web service. It is a general-purpose XML schema that can specify details of web service interfaces, bindings, and other deployment details. By having such a standard way to specify details of a service, clients who have no prior knowledge of a web service can use it.

Java EE Web Service Standards

Java API for XML-based remote procedure calls (JAX-RPC) uses an XML-based protocol for client-server remote procedure calls. JAX-RPC enables SOAP-based interoperable and portable web services. Developers use the JAX-RPC programming model to develop SOAP-based web service endpoints, along with corresponding WSDL descriptions, and clients. A JAX-RPC based web service can interact with clients that are not based on Java. Conversely, a JAX-RPC based client can interact with a non-Java-based web service implementation.

Java API for XML Web Services (JAX-WS) 2.0 is the successor to JAX-RPC that provides an “integrated stack” of APIs for developing web services, including:

- JAXB 2.0-based data binding.
- Support for the latest W3C and WS-I standards such as SOAP 1.2 and WSDL 1.2.
- Ease-of-development features using annotations.
- An improved handler framework, support for asynchronous RPC and non-HTTP transports.
- JSR 109 support.

SOAP with Attachments API for Java (SAAJ) enables developers to produce and consume messages conforming to the SOAP 1.1 specification and SOAP with Attachments note. SAAJ provides an abstraction for handling SOAP messages with attachments. Advanced developers can use SAAJ to have their applications operate directly with SOAP messages. Attachments may be complete XML documents, XML fragments, or MIME-type attachments. In addition, SAAJ allows developers to enable support for other MIME types. JAX technologies, such as JAX-RPC, internally use SAAJ to hide SOAP complexities from developers. SAAJ enables:

- Synchronous request-response messaging: the client sends a message and then waits for the response.
- One-way asynchronous messaging: the client sends a message and continues with its processing without waiting for a response.

JSR 109 (Implementing Enterprise Web Services) facilitates building interoperable web services in the Java EE environment. It standardizes the deployment of Web services in a Java EE container. JSR-109 builds upon JAX-RPC to define a standard mechanism for deploying a web service in Enterprise JavaBean (EJB) and servlet containers. Application Server supports managing JSR 109 applications based on either JAX-RPC or JAX WS 2.0.

JSR 181 (Web Services Metadata for the Java Platform) defines an annotated Java format that uses Java Language Metadata to enable easy definition of Java web services in a Java EE container.

Deploying and Testing Web Services

Application Server enables you to easily deploy and test web services. Admin Console automatically generates a test web page that makes it very easy to verify a web service's operation without writing a client application.

- [“Deploying Web Services” on page 168](#)
- [“Viewing Deployed Web Services” on page 168](#)
- [“Testing Web Services” on page 169](#)
- [“Web Services Security” on page 169](#)

Deploying Web Services

You can deploy a web service:

- In a web application, packaged as a web archive (WAR) file. Using Admin Console, go to Application Server > Applications > Web Applications, then click Deploy.
- In an enterprise Java Bean (EJB), packaged as an EJB-JAR file. Using Admin Console, go to Application Server > Applications > EJB Modules, then click Deploy.

A web service can also be implemented by a POJO (plain old Java Object). Deploy a POJO web service using the auto-deploy feature by dragging and dropping it into the auto-deploy directory. Application Server will automatically generate the appropriate deployment descriptor files and deploy the web service.

In Admin Console, you can view a list of deployed web services under Application Server > Web Services | General.

You can also use the `deploy(1)` command to deploy a web service.

Viewing Deployed Web Services

To view a list of all deployed web services with Admin Console, select Application Server > Web Services. The General tab displays a table containing the following information:

- Name of the web service. Click on it to display details.
- Application to which the web service belongs. Click on the name to display details of the web application or EJB module.
- WSDL file for the web service. Click on the file name to display the contents of the file.
- Type of the web service (SERVLET or EJB)

To view details of a web service with Admin Console, select Web Services > *web-service-name* | General. Admin Console displays the attributes of the web service:

- Name: The name of the web service.
- Endpoint Address URI: The URI of the web service endpoint.
- Application: The name of the web service application. Click on the link to display the properties of the web application or EJB module.
- WSDL: The name of the WSDL file. Click on the link to display the WSDL file for the web service.
- Module name: the name of the WAR or JAR file for the web service.
- Mapping File: The name of the mapping file. Click on the link to display the Java WSDL mapping file. This only applies to JSR 109 applications. The file will be empty for JAX-WS 2.0 applications.
- Webservices.xml: Click on the link to display the `webservices.xml` file.
- Implementation Type: SERVLET or EJB
- Implementation Class Name: Java class implementing the web service.
- Deployment Descriptors: For servlet implementations, displays `web.xml` and `sun-web.xml`; for EJB implementations, `ejb-jar.xml` and `sun-ejb-jar.xml`. Click on the links to display the files' contents.

You can also view deployed web services with the `list-components(1)` command with the `--type webservice` option.

Testing Web Services

Admin Console enables you to test web services and diagnose problems.

To test a web service with Admin Console, select Web Services > *web-service-name* | General, then click the Test button. For JAX-WS 2.0–compliant web services, Application Server generates a test page when a JAX-WS 2.0 web service is deployed. You can launch the test web page from Admin Console to easily verify a web service's operation without writing a client application.

The automatically-generated test page contains a form that enables you to invoke all the web service's methods and display the SOAP messages for each method invocation. The test page also contains a link that displays the WSDL file returned from the sever instance; that is, the runtime version of the WSDL file, not the packaged version.

Web Services Security

Application Server supports SOAP message layer security based on the SAML token profile of WS-Security. Application Server also provides tamper-proof auditing for web services. For more information, see “[Auditing Authentication and Authorization Decisions](#)” on page 96 and “[Audit Modules](#)” on page 104.

You can use default message security providers to provide web services security. Use the following commands to customize the message security providers:

- `create-message-security-provider(1)`
- `delete-message-security-provider(1)`
- `list-message-security-providers(1)`

For more information about configuring security for web services, see [Chapter 9, “Configuring Message Security”](#)

Publishing to Web Services Registries

A web service registry is like a “phone book” for web services. It enables client applications to look up and find web services. Application Server supports two widely-used registry standards: ebXML (Electronic Business using XML) and Universal Description, Discovery, and Integration (UDDI). A company can host an internal registry for use by web services within the corporate firewall, or externally for use by customers, business partners, and other external entities.

Note – Application Server does not have an internal registry. To publish web services to an internal registry, you must download and install the registry on the application server. To publish a web service to an external registry, specify the address of the external registry.

Adding a Connector Module for a Registry

Application Server has client-side implementations to enable you to communicate with registry servers plugged in as connector resource adapters. This enables Application Server to transparently connect to multiple registry types by using different resource adapters.

Before you can add a registry to which to publish web services, you must deploy an appropriate connector module. With Admin Console, add a connector module at Application Server > Applications > Connector Modules. Click the Deploy button to deploy a new connector module, then choose the appropriate resource adapter archive (RAR) file.

Application Server comes with a pre-deployed resource adapter for a UDDI registry. So, you do not need to deploy a resource adapter to use a UDDI registry.

Setting up a Registry for use with Application Server

Application Server does not currently provide a web services registry. If you want to use an ebXML registry, download the Java Web Services Developer Pack (JWSDP) 1.6, which includes an ebXML registry. To run JWSDP, you must use Application Server 8.2 or another web container such as Tomcat.

To set up your own registry to work with Application Server, follow the procedures in this section.

▼ To Create a Registry

- 1 **Download Java Web Services Developer Pack (JWSDP) 1.6:**
<http://java.sun.com/webservices/downloads/1.6/index.html>.
- 2 **Install JWSDP 1.6 on your system.**
- 3 **Download and install the desired web container.**
 JWSDP 1.6 works with Application Server 8.2, J2EE 1.4 SDK, and Tomcat 5.0 for JWSDP. You cannot use Application Server 9 as the web container with JWSDP 1.6.
- 4 **Change the default port that the web container uses.**
 Application Server 9 uses port 8080 by default, so you must change the port that the web container uses to avoid conflicts. Change the port, for example, to 7080. With Tomcat, you can do this in the `server.xml` file. In Sun Java System Application Server 8.2 you can do this using Admin Console.
- 5 **Start the container.**
 This starts the installed registry by default. JWSDP 1.6 comes with the registry preconfigured.
- 6 **Configure the resource adapter to work with Application Server, as described in the next section.**

▼ To Configure a Resource Adapter for an ebXML Registry

The result of this procedure will be to create a `soar.rar` resource adapter file that you can deploy to Application Server.

Before You Begin Set the `JWSDP_HOME` environment variable to the directory where JWSDP 1.6 is installed and the `JAVA_HOME` environment variable to the directory where J2SE is installed.

- 1 **Download the `ra.xml` file from the Glassfish site:**
<https://glassfish.dev.java.net/javaee5/ws-mgmt/registry/ra.xml>.
 Save the file in the `/tmp` directory (`C:\tmp` on Windows).
- 2 **You can either download and execute a script, or perform the steps manually.**
 - **To use a script:**
 - a. **Download one of the following scripts:**

- [Shell script for Solaris and Linux.](#)
- [BAT file for Windows.](#)

Save the script to the /tmp directory (C:\tmp on Windows).

b. Execute the script.

- **To perform the steps manually:**

a. Enter the following commands:

```
cd $JWSDP_HOME/registry/lib
mkdir tmp
cp soar-jaxr-ra.jar ./tmp
cd tmp
jar xvf soar-jaxr-ra.jar
rm soar-jaxr-ra.jar ra.xml ./META-INF/SUN* ./META-INF/pack.properties
cp ra.xml META-INF .
cp $JWSDP_HOME/registry/lib/oasis-* .
cp $JWSDP_HOME/registry/lib/omar-common.jar .
cp $JWSDP_HOME/registry/lib/jaxr-ebxml.jar .
cp $JWSDP_HOME/jwsdp-shared/lib/commons-logging.jar .
```

Note – The commands shown are for Unix or Linux. Use the equivalent commands on Windows.

Now, the tmp directory should contain the following files and directories:

- META-INF directory
- com directory
- jaxr-ebxml.jar
- oasis-regrep.jar
- omar-common.jar
- oasis-saml1.1.jar
- oasis-saml2.0.jar
- commons-logging.jar

b. Enter this command to create resource adapter file soar.rar:

```
jar cvf soar.rar META-INF com jaxr-ebxml.jar oasis-regrep.jar omar-common.jar oasis-saml1.1.jar oasis-saml2.0.jar commons-logging.jar
```

Note – To use the jar command on Windows, you must have the JAVA_HOME\bin directory on your path.

Next Steps Once you have created the soar.rar file, deploy it to create a registry as described in the next section.

Adding a Registry

Adding a registry is equivalent to creating a connector connection pool and adding a corresponding resource to the Application Server. This functionality is provided as a convenience, and enables Application Server to communicate to a registry, as described previously.

Add or remove a web services registry with Admin Console at Application Server > Web Services | Registry. Click Add to add a new registry to which to publish web services. The Add Registry Page appears. Use this page to create a Registry Access Point (RAP). For help on using Admin Console, click Help.

When you add a registry, specify the following parameters:

- JNDI Name: the connection resource pool (JNDI) name of the registry. The JNDI Name of this connector resource is the JNDI Name of the registry.
- Choose the type of the registry to add: UDDI 3.0 or ebXML.
- Publish URL and Query URL: the addresses for publishing and querying the registry, respectively. The format is: `http://hostname/path-to-registry—installation`. For example, `http://localhost:7080/soar/registry/soap`.
- User name and password for the registry (not required for ebXML; fields are disabled).

Publishing a Web Service to a Registry

To publish a web service with Admin Console, select Web Services > *web-service-name* | Publish.

In the Publish Web Service screen, select one or more registries to which you want to publish the web service, then click Publish. To publish to all the available registries, click the Add All button.

Enter categories under which this web service will show up in the registry. Use a comma to separate each category. The categories are defined in the registry you are using. Enter a description and the name of the organization for this web service.

If you are using a load balancer, enter the load balancer host name, port number, and the SSL port number. If you are publishing the web service to an external registry, where the WSDL can be found over the internet, these options will replace the hostname and port name specified in the WSDL to the one of the load balancer.

To un-publish a web service, In the Publish Web Service screen, select the registry from which you want to un-publish the web service, then click Unpublish.

You can also use the following commands to publish a web service, un-publish a web service, and list registry locations:

- [publish-to-registry\(1\)](#)
- [unpublish-from-registry\(1\)](#)
- [list-registry-locations\(1\)](#)

Monitoring Web Services

Admin Console can track operational statistics for web services, and can display messages sent and received by web services.

To enable monitoring for a web service, with Admin Console, select Web Services > *web-service-name* | Monitor | Configuration.

In the Monitoring Configuration page, set the monitoring level:

- **LOW:** Monitors response time, throughput, total number of requests, faults, average response time, and response time for last request to the web service.
- **HIGH:** In addition to the metrics monitored at LOW level, also monitors SOAP request and response messages.
- **OFF:** Disables monitoring.

Enter a value for the Message History. The default is 25. Click the Reset button to clear all statistics and the running averages are restarted.

You can also configure web service monitoring with the [configure-webservice-management\(1\)](#) command.

Viewing Web Service Statistics

Application Server9 provides capabilities to track the operational statistics of a web service.

View monitoring statistics at Web Services > *web-service-name* | Monitor | Statistics. The statistics available are:

- Response time in milliseconds on any successful or unsuccessful operation (maximum, minimum, and average).

Note – Response time is the time the sever took to process the web service request, not including any network transmission time.

- Throughput.
- Total number of requests.
- Total number of faults.

Viewing Web Service Messages

You can also configure a web service to view messages for a web service endpoint. Monitor web service messages at Web Services > *web-service-name* | Monitor | Messages.

By default, Admin Console retains the last 25 messages. Admin Console displays details of SOAP requests, responses, and HTTP header information. In addition, the following information is displayed for a web service request:

- Remote user name
- Client host information
- Timestamp
- Response information (success or failure)
- Size of the request

You can also select a filter to view only the success messages or the failure messages. If call flow monitoring is enabled, each message is linked to call flow information that enables you to look at the call stack information for each message.

Transforming Messages with XSLT Filters

You can apply XSLT transformation rules to a web service end point. This enables fine-grained control of web service requests and responses. You can apply multiple XSLT rules to a web service end point method, and you can configure the order in which you apply the transformations.

You can apply transformation rule to a SOAP request or response.

Before revising an application to apply transformation rules, evaluate the pros and cons of doing so: Applying XSLT can lengthen processing time for a web service. In other cases, XSLT makes sense, for example if a web service is exposed to partners with different XML data exchange formats or different security requirements. In such cases, transformation rules can act as a proxy for the web service and be advantageous.

To add a transformation rule to apply to a web service operation with Admin Console, select Web Services > *web-service-name* | Transformation. Click Add. Admin Console displays a list of transformation rules available for the web service endpoint. To add a new transformation rule, click Add, and then browse to the location of the XSLT file that contains the transformation rule.

To enable a transformation rule, in the Transformation Rules page select the check box corresponding to the rule, then click Enable. To disable the a rule, click Disable.

If you add multiple transformation rules for a web service endpoint, the transformation rules are applied in the order in which they are added.

To remove a transformation rule, in the Transformation Rules page select the check box corresponding to the rule, then click Remove. This removes the transformation rule from the list and removes the corresponding XSLT file.

Additionally, you can create, delete, and list transformation rules with the following `asadmin` commands:

- `create-transformation-rule(1)`
- `delete-transformation-rule(1)`
- `list-transformation-rules(1)`

Web Services as JBI Service Providers

By default, Application Server exposes web services as Java Business Integration (JBI) service providers for Open Enterprise Service Bus (ESB). Afterwards, flexible protocol binding and the components in the ESB can directly communicate with Java EE applications, using, for example, JMS.

You can disable this feature by using `asadmin` to set the endpoint's `jbi-enabled` property to `false`. For example, to deactivate `HelloImpl` from the JBI environment, use the commands:

```
asadmin configure-webservice-management server_HelloImpl#HelloImpl
asadmin set server.applications.web-module.server_HelloImpl.webservice-endpoint.HelloImpl.jbi-enabled=false
```

Open ESB implements an Enterprise Service Bus (ESB) runtime with sample service engines and binding components. Open ESB enables you to create composite applications by loosely coupling enterprise applications and web services. Additionally, you can easily recompose such composite applications, realizing the benefits of a true service oriented architecture (SOA). The core of the Open ESB SDK is based on Java Business Integration (JBI) technology, defined in the JSR 208 specification.

For more information on JBI and Open ESB, see [Project Open ESB](#).

Configuring the Object Request Broker

This section describes how to configure the Object Request Broker (ORB) and IIOP listeners. It has the following topics:

- [“About the Object Request Broker” on page 177](#)
- [“Configuring the ORB” on page 178](#)
- [“Working with IIOP Listeners” on page 178](#)

About the Object Request Broker

- [“CORBA” on page 177](#)
- [“What is the ORB?” on page 178](#)
- [“IIOP Listeners” on page 178](#)

CORBA

The Application Server supports a standard set of protocols and formats that ensure interoperability. Among these protocols are those defined by CORBA.

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 the operation that needs to be performed, including the object name (called an object reference) of the service provider and parameters, if any, for the invoked method. CORBA automatically handles network programming tasks such as object registration, object location, object activation, request de-multiplexing, error-handling, marshalling, and operation dispatching.

What is the ORB?

The Object Request Broker (ORB) 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.

A 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 the Internet Inter-Orb Protocol (IIOP for short). The remote ORB then locates the appropriate object, processes the request, and returns the results.

IIOP can be used as a Remote Method Invocation (RMI) protocol by applications or objects using RMI-IIOP. Remote clients of enterprise beans (EJB modules) communicate with the Application Server via RMI-IIOP.

IIOP Listeners

An IIOP listener is a listen socket that accepts incoming connections from the remote clients of enterprise beans and from other CORBA-based clients. Multiple IIOP listeners can be configured for the Application Server. For each listener, specify a port number, a network address, and optionally, security attributes.

Configuring the ORB

To configure the ORB, click Configuration > ORB in the Admin Console and provide values for the fields. After saving your changes, restart the server.

Working with IIOP Listeners

To create an IIOP listener, select Configuration > ORB > IIOP Listeners and click New in the Admin Console. Alternatively, you can use the following `asadmin` command to create IIOP listeners: `create-iiop-listener(1)` and `create-ssl(1)`.

To edit an IIOP listener, select Configuration > ORB > IIOP Listeners and select the listener to be modified in the Admin Console. Modify the settings. If you have changed the port number, restart the server.

To delete an IIOP listener, select Configuration > ORB > IIOP Listeners and select the listener to be deleted in the Admin Console. Alternatively, you can use the `delete-iiop-listener(1)` command.

Thread Pools

This chapter describes how to create, edit, and delete thread pools. It has the following sections:

- [“About Thread Pools” on page 179](#)
- [“Configuring Thread Pools” on page 180](#)

About Thread Pools

This section describes thread pools and how they work in the Application Server.

The Java Virtual Machine (JVM) can support many threads of execution at once. To help performance, the Application Server maintains one or more thread pools. It is possible to assign specific thread pools to connector modules and to the ORB.

One thread pool can serve multiple connector modules and enterprise beans. Request threads handle user requests for application components. When the server receives a request, it assigns the request to a free thread from the thread pool. The thread executes the client's requests and returns results. For example, if the request needs to use a system resource that is currently busy, the thread waits until that resource is free before allowing the request to use that resource.

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 that is specified signals the server to allocate at least that many threads in reserve for application requests. That number is increased up to the maximum thread pool size that is specified.

Increasing the number of threads available to a process allows the process to respond to more application requests simultaneously.

Avoid thread starvation, where one resource adapter or application occupies all threads in the Application Server, by dividing the Application Server threads into different thread-pools.

See Also:

- [“Creating a Thread Pool” on page 180](#)
- [“Editing the Thread Pool Settings” on page 181](#)
- [“Deleting a Thread Pool” on page 181](#)
- [“Configuring the ORB” on page 178](#)

Configuring Thread Pools

- [“Creating a Thread Pool” on page 180](#)
- [“Editing the Thread Pool Settings” on page 181](#)
- [“Deleting a Thread Pool” on page 181](#)

Creating a Thread Pool

To create a thread pool using the Admin Console, go to Configuration > Thread Pools > Current Pools > New.

- Enter the name of the thread pool in the Thread Pool ID field.
- Enter the minimum number of threads in the thread pool servicing requests in this queue in the Minimum Thread Pool Size field.
These threads are created up front when this thread pool is instantiated.
- Enter the maximum number of threads in the thread pool servicing requests in this queue in the Maximum Thread Pool Size field.
This is the upper limit on the number of threads that exist in the thread pool.
- Enter the number, in seconds, after which idle threads are removed from pool in the Idle Timeout field.
- Enter the total number of work queues that are serviced by this thread pool in the Number of Work Queues field.
- Restart the Application Server.

For more details on creating thread pools, click Help in the Admin Console.

You can also create a thread pool from the command line by using the `asadmin` command, [create-threadpool\(1\)](#).

See Also:

- [“About Thread Pools” on page 179](#)
- [“Editing the Thread Pool Settings” on page 181](#)
- [“Deleting a Thread Pool” on page 181](#)

Editing the Thread Pool Settings

To edit a settings for a thread pool using the Admin Console, go to Configuration > Thread Pools > Current Pools and select the pool you want to configure. Modify the values for the selected thread pool, save and restart the Application Server.

For more details on editing thread pools, click Help in the Admin Console.

Deleting a Thread Pool

To delete a thread pool using the Admin Console, go to Configuration > Thread Pools > Current Pools. Check the thread pool name to be deleted and click Delete.

Restart the Application Server.

You can also create a thread pool from the command line by using the `asadmin` command, `delete-threadpool(1)`.

See Also:

- [“About Thread Pools” on page 179](#)
- [“Creating a Thread Pool” on page 180](#)
- [“Editing the Thread Pool Settings” on page 181](#)

Configuring Logging

This chapter briefly describes how to use the Admin Console to configure logging and view the server log. This chapter contains the following sections:

- [“About Logging” on page 183](#)
- [“Configuring Logging” on page 186](#)

About Logging

- [“Log Records” on page 183](#)
- [“The Logger Namespace Hierarchy” on page 184](#)

Log Records

The Application Server uses the Java EE 5 platform Logging API specified in JSR 047. Application Server logging messages are recorded in the server log, normally found at *domain-dir/logs/server.log*.

The *domain-dir/logs* directory contains two other kinds of logs in addition to the server log. In the *access* subdirectory are the HTTP Service access logs, and in the *tx* subdirectory are the Transaction Service logs. For information about these logs, click Help in the Admin Console.

The components of the Application Server generate logging output. Application components can also generate logging output.

Application components may use the Apache Commons Logging Library to log messages. The platform standard JSR 047 API, however, is recommended for better log configuration.

Log records follow a uniform format:

```
[#|yyyy-mm-ddThh:mm:ss.SSS-Z|Log Level|ProductName-Version|LoggerName|Key Value Pairs|Message|#]
```

For example:

```
[#|2004-10-21T13:25:53.852-0400|INFO|sun-appserver-epe8.1|javax.enterprise.  
system.core|_ThreadID=13;|CORE5004: Resource Deployed:  
[cr:jms/DurableConnectionFactory].|#]
```

In this example,

- `[#` and `#]` mark the beginning and end of the record.
- The vertical bar (`|`) separates the record fields.
- `2004-10-21T13:25:53.852-0400` specifies the date and time.
- The *Log Level* is `INFO`. This level may have any of the following values: `SEVERE`, `WARNING`, `INFO`, `CONFIG`, `FINE`, `FINER`, and `FINEST`.
- The *ProductName-Version* is `sun-appserver-pe8.1`.
- The *LoggerName* is a hierarchical logger namespace that identifies the source of the log module, in this case `javax.enterprise.system.core`.
- The *Key Value Pairs* are key names and values, typically a thread ID such as `_ThreadID=14;`.
- The *Message* is the text of the log message. For all Application Server `SEVERE` and `WARNING` messages and many `INFO` messages, it begins with a message ID that consists of a module code and a numerical value (in this case, `CORE5004`).

The log record format might be changed or enhanced in future releases.

See Also:

- [“The Logger Namespace Hierarchy” on page 184](#)

The Logger Namespace Hierarchy

The Application Server provides a logger for each of its modules. The following table lists the names of the modules and the namespace for each logger in alphabetical order, as they appear on the Log Levels page of the Admin Console (see [“Configuring Log Levels” on page 186](#)). The last three modules in the table do not appear on the Log Levels page.

TABLE 15-1 Application Server Logger Namespaces

Module Name	Namespace
Admin	<code>javax.enterprise.system.tools.admin</code>
ClassLoader	<code>javax.enterprise.system.core.classloading</code>
Configuration	<code>javax.enterprise.system.core.config</code>

TABLE 15-1 Application Server Logger Namespaces (Continued)

Module Name	Namespace
Connector	<code>javax.enterprise.resource.resourceadapter</code>
CORBA	<code>javax.enterprise.resource.corba</code>
Deployment	<code>javax.enterprise.system.tools.deployment</code>
EJB Container	<code>javax.enterprise.system.container.ejb</code>
JavaMail	<code>javax.enterprise.resource.javamail</code>
JAXR	<code>javax.enterprise.resource.webservices.registry</code>
JAXRPC	<code>javax.enterprise.resource.webservices.rpc</code>
JMS	<code>javax.enterprise.resource.jms</code>
JTA	<code>javax.enterprise.resource.jta</code>
JTS	<code>javax.enterprise.system.core.transaction</code>
MDB Container	<code>javax.enterprise.system.container.ejb.mdb</code>
Naming	<code>javax.enterprise.system.core.naming</code>
Persistence	<code>oracle.toplink.essentials,</code> <code>javax.enterprise.resource.jdo,</code> <code>javax.enterprise.system.container.cmp</code>
Node Agent (Enterprise Edition only)	<code>javax.ee.enterprise.system.nodeagent</code>
Root	<code>javax.enterprise</code>
SAAJ	<code>javax.enterprise.resource.webservices.saaaj</code>
Security	<code>javax.enterprise.system.core.security</code>
Self Management	<code>javax.enterprise.system.core.selfmanagement</code>
Server	<code>javax.enterprise.system</code>
Synchronization (Enterprise Edition only)	<code>javax.ee.enterprise.system.tools.synchronization</code>
Util	<code>javax.enterprise.system.util</code>
Verifier	<code>javax.enterprise.system.tools.verifier</code>
Web Container	<code>javax.enterprise.system.container.web</code>
Core	<code>javax.enterprise.system.core</code>
System Output (<code>System.out.println</code>)	<code>javax.enterprise.system.stream.out</code>
System Error (<code>System.err.println</code>)	<code>javax.enterprise.system.stream.err</code>

See Also:

- [“Log Records” on page 183](#)
- [“Configuring Log Levels” on page 186](#)

Configuring Logging

- [“Configuring General Logging Settings” on page 186](#)
- [“Configuring Log Levels” on page 186](#)
- [“Viewing Server Logs” on page 187](#)

Configuring General Logging Settings

To configure the general logging settings using the Admin Console, go to Application Server > Logging> Logging Settings > General. On the General page, enter appropriate values to customize logging to your requirements. Stop and restart the Application Server

For details on setting the various configuration parameters, click Help in the Admin Console.

See Also:

- [“Configuring Log Levels” on page 186](#)
- [“Viewing Server Logs” on page 187](#)
- [“Log Records” on page 183](#)
- [“The Logger Namespace Hierarchy” on page 184](#)

Configuring Log Levels

To configure log levels using the Admin Console, go to Application Server > Logging> Logging Settings > Log Levels. Set the log level for the modules listed on this page. Use the Additional Properties area to configure log levels for any application loggers. For a list of the module loggers, see [“The Logger Namespace Hierarchy” on page 184](#).

For details on setting the various configuration parameters, click Help in the Admin Console.

- [“Configuring General Logging Settings” on page 186](#)
- [“Viewing Server Logs” on page 187](#)
- [“Log Records” on page 183](#)
- [“The Logger Namespace Hierarchy” on page 184](#)

Viewing Server Logs

There are two methods to view the Application Server's log files:

- Go to Common Tasks node, then click Search Log files.
- or,
- Go to Application Server > Logging > View Log Files.

Use the options provided in the Search Criteria area to display log results based on your preferences.

- **Instance Name** — Choose an instance name from the drop-down list to view the log for that server instance. The default is the current server instance.
- **Log File** — Choose a log file name from the drop-down list to view the contents of that log. The default is `server.log`.
- **Timestamp** — To view the most recent messages, select Most Recent (the default). To view messages only from a certain period of time, select Specific Range and type a date and time value in the From and To fields that appear. For the Time value, the syntax must take the following form (SSS stands for milliseconds):

`hh:mm:ss.SSS`

For example:

`17:10:00.000`

If the From value is later than the To value, an error message appears.

- **Log Level** — To filter messages by log level, choose a log level from the drop-down list. By default, the display includes all messages that appear in the server log at the chosen log level and more severe levels. Select the checkbox labeled “Do not include more severe messages” to display messages at only the chosen level.

To ensure that the messages you want to view appear in the server log, first set the appropriate log levels on the Log Levels page. See [“Configuring Log Levels” on page 186](#)

If you choose to filter log messages based on log level, only messages matching the specified filter criteria are shown. However, this filtering does not affect which messages are logged to the server log.

The most recent 40 entries in the server log appear, with the settings specified on the Logging Settings and Log Levels pages.

Click the triangle next to the Timestamp header to sort the messages so that the most recent one appears last.

To view a formatted version of any message, click the link marked

(details)

A window labeled Log Entry Detail appears, with a formatted version of the message.

At the end of the list of entries, click the buttons to view earlier or later entries in the log file.

Click Advanced Search in the Search Criteria area to make additional refinements to the log viewer. Use the Advanced Options fields as follows:

- **Logger** — To filter by module, choose one or more namespaces from the drop-down list. Use shift-click or control-click to choose multiple namespaces.
Selecting a namespace at a higher level selects all the namespaces below it. For example, selecting `javax.enterprise.system` also selects the loggers for all the modules under that namespace: `javax.enterprise.system.core`, `javax.enterprise.system.tools.admin`, and so on.
- **Custom Logger** — To view messages from loggers specific to a particular application, type the logger names in the text field, one per line. If the application has several modules, you can view any or all of them. For example, suppose the application has loggers with the following names:

```
com.mycompany.myapp.module1
com.mycompany.myapp.module2
com.mycompany.myapp.module3
```

To view messages from all modules in the application, type `com.mycompany.myapp`. To view messages from `module2` only, type `com.mycompany.myapp.module2`.

When you specify one or more custom loggers, messages from Application Server modules appear only if you specify them explicitly in the Logger area.

- **Name-Value Pairs** — To view output from a specific thread, type the key name and value for that thread in the text field. The key name is `_ThreadID`. For example:

```
_ThreadID=13
```

Suppose that `com.mycompany.myapp.module2` runs in several threads. To refine the log viewer to show only the output from a single thread, specify that module's logger in the Custom Logger field, and then specify the thread ID in this field.

- **Display** — To view more than 40 messages at a time (the default), choose another of the available values from the drop-down list (100, 250, or 1000).

To view stack traces, deselect the “Limit excessively long messages” checkbox. By default, stack traces do not appear in the viewer; to view them, click the `(details)` link for a message.

Click Basic Search to hide the Advanced Options area.

See Also:

- [“Log Records” on page 183](#)

- [“The Logger Namespace Hierarchy” on page 184](#)
- [“Configuring General Logging Settings” on page 186](#)
- [“Configuring Log Levels” on page 186](#)

Monitoring Components and Services

This chapter contains information about monitoring components using the Application Server Admin Console. This chapter contains the following sections:

- [“About Monitoring” on page 191](#)
- [“Enabling and Disabling Monitoring” on page 212](#)
- [“Viewing Monitoring Data” on page 214](#)
- [“Using JConsole” on page 229](#)

About Monitoring

- [“Monitoring in the Application Server” on page 191](#)
- [“Overview of Monitoring” on page 192](#)
- [“About the Tree Structure of Monitorable Objects” on page 192](#)
- [“About Statistics for Monitored Components and Services” on page 196](#)

Monitoring in the Application Server

Use monitoring to observe the runtime state of various components and services deployed in a server instance of the Application Server. With the information on the state of runtime components and processes, it is possible to identify performance bottlenecks for tuning purposes, aid capacity planning, predict failures, do root cause analysis in case of failures, and ensure that everything is functioning as expected.

Turning monitoring on reduces performance by increasing overhead.

See Also:

- [“Overview of Monitoring” on page 192](#)

Overview of Monitoring

To monitor the Application Server, perform these steps:

1. Enable the monitoring of specific services and components using either the Admin Console or the `asadmin` tool.

For more information on this step, refer to [“To configure monitoring levels using the Admin Console” on page 212](#).

For more information on this step, refer to [“Enabling and Disabling Monitoring” on page 212](#).

2. View monitoring data for the specified services or components using either the Admin Console or the `asadmin` tool.

For more information on this step, refer to [“Viewing monitoring data in the Admin Console” on page 214](#) or [“Viewing Monitoring Data With the `asadmin` Tool” on page 214](#).

For more information on this step, refer to [“Viewing Monitoring Data” on page 214](#).

See Also:

- [“To configure monitoring levels using the Admin Console” on page 212](#)
- [“Viewing monitoring data in the Admin Console” on page 214](#)
- [“Viewing Monitoring Data With the `asadmin` Tool” on page 214](#)

About the Tree Structure of Monitorable Objects

The Application Server uses a tree structure to track monitorable objects. Because the tree of monitoring objects is dynamic, it changes as components are added, updated, or removed in the instance. The root object in the tree is the server instance name, for example, `server`. (In the Platform Edition, just one server instance is permitted.)

The following command displays the top level of the tree:

```
asadmin> list --user adminuser --monitor server
server.applications
server.http-service
server.connector-service
server.jms-service
server.jvm
server.orb
server.resources
server.thread-pools
```

The following sections describe these sub-trees:

- [“The Applications Tree” on page 193](#)
- [“The HTTP Service Tree” on page 194](#)

- “The Resources Tree” on page 194
- “The Connector Service Tree” on page 194
- “The JMS Service Tree” on page 195
- “The ORB Tree” on page 195
- “The Thread Pool Tree” on page 195

The Applications Tree

The following schematic shows the top and child nodes for the various components of enterprise applications. The nodes at which monitoring statistics are available are marked with an asterisk (*). For more information, refer to “EJB Container Statistics” on page 196.

EXAMPLE 16-1 Applications Node Tree Structure

```

applications
|--- application1
|   |--- ejb-module-1
|   |   |--- ejb1 *
|   |       |--- cache (for entity/sfsb) *
|   |       |--- pool (for slsb/mdb/entity) *
|   |       |--- methods
|   |           |---method1 *
|   |           |---method2 *
|   |           |--- stateful-session-store (for sfsb)*
|   |           |--- timers (for slsb/entity/mdb) *
|   |--- web-module-1
|   |   |--- virtual-server-1 *
|   |       |---servlet1 *
|   |       |---servlet2 *
|--- standalone-web-module-1
|   |   |----- virtual-server-2 *
|   |       |---servlet3 *
|   |       |---servlet4 *
|   |   |----- virtual-server-3 *
|   |       |---servlet3 *(same servlet on different vs)
|   |       |---servlet5 *
|--- standalone-ejb-module-1
|   |   |--- ejb2 *
|   |       |--- cache (for entity/sfsb) *
|   |       |--- pool (for slsb/mdb/entity) *
|   |       |--- methods
|   |           |--- method1 *
|   |           |--- method2 *
|--- application2

```

The HTTP Service Tree

The nodes of the HTTP service are shown in the following schematic. The nodes at which monitoring information is available are marked with an asterisk (*). See [“HTTP Service Statistics” on page 202](#).

EXAMPLE 16-2 HTTP Service Schematic (Platform Edition version)

```
http-service
|--- virtual-server-1
|   |--- http-listener-1 *
|   |--- http-listener-2 *
|--- virtual-server-2
|   |--- http-listener-1 *
|   |--- http-listener-2 *
```

EXAMPLE 16-3 HTTP Service Schematic (Enterprise Edition version)

```
http-service *
|--- connection-queue *
|--- dns *
|--- file-cache *
|--- keep-alive *
|--- pwc-thread-pool *
|--- virtual-server-1*
|       |--- request *
|--- virtual-server-2*
|       |--- request *
```

The Resources Tree

The resources node holds monitorable attributes for pools such as the JDBC connection pool and connector connection pool. The following schematic shows the top and child nodes for the various resource components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See [“JDBC Connection Pools Statistics” on page 203](#).

EXAMPLE 16-4 Resources Schematic

```
resources
|--- connection-pool1(either connector-connection-pool or jdbc)*
|--- connection-pool2(either connector-connection-pool or jdbc)*
```

The Connector Service Tree

The connector services node holds monitorable attributes for pools such as the connector connection pool. The following schematic shows the top and child nodes for the various

connector service components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See [“JMS/Connector Service Statistics” on page 204](#).

EXAMPLE 16-5 Connector Service Schematic

```
connector-service
|--- resource-adapter-1
|       |-- connection-pools
|       |       |-- pool-1 (All pool stats for this pool)
|       |-- work-management (All work mgmt stats for this RA)
```

The JMS Service Tree

The JMS services node holds monitorable attributes for pools such as the connector connection pool. The following schematic shows the top and child nodes for the various JMS service components. The nodes at which monitoring statistics are available are marked with an asterisk (*).

EXAMPLE 16-6 JMS Service Schematic

```
jms-service
|-- connection-factories [AKA conn. pools in the RA world]
|       |-- connection-factory-1 (All CF stats for this CF)
|-- work-management (All work mgmt stats for the MQ-RA)
```

The ORB Tree

The ORB node holds monitorable attributes for connection managers. The following schematic shows the top and child nodes for the ORB components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See [“Statistics for Connection Managers in an ORB” on page 206](#).

EXAMPLE 16-7 ORB Schematic

```
orb
|--- connection-managers
|       |-- connection-manager-1 *
|       |-- connection-manager-1 *
```

The Thread Pool Tree

The thread pool node holds monitorable attributes for connection managers. The following schematic shows the top and child nodes for the ORB components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See [“Thread Pools Statistics” on page 206](#).

EXAMPLE 16-8 Thread Pool Schematic

```
thread-pools
|   |--- thread-pool-1 *
|   |--- thread-pool-2 *
```

About Statistics for Monitored Components and Services

This section describes the monitoring statistics that are available:

- “EJB Container Statistics” on page 196
- “Web Container Statistics” on page 200
- “HTTP Service Statistics” on page 202
- “JDBC Connection Pools Statistics” on page 203
- “JMS/Connector Service Statistics” on page 204
- “Statistics for Connection Managers in an ORB” on page 206
- “Thread Pools Statistics” on page 206
- “Transaction Service Statistics” on page 207
- “Java Virtual Machine (JVM) Statistics” on page 207
- “JVM Statistics in J2SE 5.0” on page 208

EJB Container Statistics

EJB statistics are described in the following table.

TABLE 16-1 EJB Statistics

Attribute Name	Data Type	Description
createcount	CountStatistic	Number of times an EJB’s create method is called.
removecount	CountStatistic	Number of times an EJB’s remove method is called.
pooledcount	RangeStatistic	Number of entity beans in pooled state.
readycount	RangeStatistic	Number of entity beans in ready state.
messagecount	CountStatistic	Number of messages received for a message-driven bean.

TABLE 16-1 EJB Statistics (Continued)

Attribute Name	Data Type	Description
methodreadycount	RangeStatistic	Number of stateful or stateless session beans that are in the MethodReady state.
passivecount	RangeStatistic	Number of stateful session beans that are in Passive state.

The statistics available for EJB method invocations are listed in the following table.

TABLE 16-2 EJB Method Statistics

Attribute Name	Data Type	Description
methodstatistic	TimeStatistic	Number of times an operation is called; the total time that is spent during the invocation, and so on.
totalnumerrors	CountStatistic	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 for the EJB container.
totalnumsuccess	CountStatistic	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.
executiontime	CountStatistic	Time (ms) spent executing the method for the last successful/unsuccessful attempt to execute the operation. This is collected for stateless and stateful session beans and entity beans if monitoring is enabled on the EJB container.

The statistics for EJB Session Stores are listed in the following table.

TABLE 16-3 EJB Session Store Statistics

Attribute Name	Data Type	Description
currentSize	RangeStatistic	Number of passivated or checkpointed sessions currently in the store.
activationCount	CountStatistic	Number of sessions activated from the store.
activationSuccessCount	CountStatistic	Number of sessions successfully activated from the store
activationErrorCount	CountStatistic	Time (ms) spent executing the method for the last successful/unsuccessful attempt to execute the operation. This is collected for stateless and stateful session beans and entity beans, if monitoring is enabled on the EJB container.
passivationCount	CountStatistic	Number of sessions passivated (inactivated) using this store.
passivationSuccessCount	CountStatistic	Number of sessions successfully passivated using this store.
passivationErrorCount	CountStatistic	Number of sessions that could not be passivated using this store.
expiredSessionCount	CountStatistic	Number of expired sessions that were removed by this store.
passivatedBeanSize	CountStatistic	Total number of bytes passivated by this store, including total, minimum, and maximum.
passivationTime	CountStatistic	Time spent on passivating beans to the store, including the total, minimum, and maximum.
checkpointCount (EE only)	CountStatistic	Number of sessions checkpointed using this store.
checkpointSuccessCount (EE only)	CountStatistic	Number of sessions checkpointed successfully.
checkpointErrorCount (EE only)	CountStatistic	Number of sessions that couldn't be checkpointed.

TABLE 16-3 EJB Session Store Statistics (Continued)

Attribute Name	Data Type	Description
checkpointedBeanSize (EE only)	ValueStatistic	Total number of beans checkpointed by the store.
checkpointTime (EE only)	TimeStatistic	Time spent on checkpointing beans to the store.

The statistics available for EJB pools are listed in the following table.

TABLE 16-4 EJB Pool Statistics

Attribute Name	Data Type	Description
numbeansinpool	BoundedRangeStatistic	Number of EJB's in the associated pool, providing an idea about how the pool is changing.
numthreadswaiting	BoundedRangeStatistic	Number of threads waiting for free beans, giving an indication of possible congestion of requests.
totalbeanscreated	CountStatistic	Number of beans created in associated pool since the gathering of data started.
totalbeansdestroyed	CountStatistic	Number of beans destroyed from associated pool since the gathering of data started.
jmsmaxmessagesload	CountStatistic	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.

The statistics available for EJB caches are listed in the following table.

TABLE 16-5 EJB Cache Statistics

Attribute Name	Data Type	Description
cachemisses	BoundedRangeStatistic	The number of times a user request does not find a bean in the cache.
cachehits	BoundedRangeStatistic	The number of times a user request found an entry in the cache.

TABLE 16–5 EJB Cache Statistics (Continued)

Attribute Name	Data Type	Description
numbeansincache	BoundedRangeStatistic	The number of beans in the cache. This is the current size of the cache.
numpassivations	CountStatistic	Number of passivated beans. Applies only to stateful session beans.
numpassivationerrors	CountStatistic	Number of errors during passivation. Applies only to stateful session beans.
numexpiredsessionsremoved	CountStatistic	Number of expired sessions removed by the cleanup thread. Applies only to stateful session beans.
numpassivationsuccess	CountStatistic	Number of times passivation completed successfully. Applies only to stateful session beans.

The statistics available for Timers are listed in the following table.

TABLE 16–6 Timer Statistics

Statistic	Data Type	Description
numtimerscreated	CountStatistic	Number of timers created in the system.
numtimersdelivered	CountStatistic	Number of timers delivered by the system.
numtimersremoved	CountStatistic	Number of timers removed from the system.

Web Container Statistics

The web container fits into the tree of objects as shown in [“The Applications Tree” on page 193](#). Web container statistics are displayed for each individual web application. Statistics available for the web container for Servlets are shown in [“Web Container Statistics” on page 200](#) and statistics available for web modules are shown in [“Web Container Statistics” on page 200](#).

TABLE 16–7 Web Container (Servlet) Statistics

Statistic	Units	Data Type	Comments
errorcount	Number	CountStatistic	Cumulative number of cases where the response code is greater than or equal to 400.
maxtime	Milliseconds	CountStatistic	The maximum amount of time the web container waits for requests.

TABLE 16-7 Web Container (Servlet) Statistics (Continued)

Statistic	Units	Data Type	Comments
processingtime	Milliseconds	CountStatistic	Cumulative value of the amount of time required to process each request. The processing time is the average of request processing times divided by the request count.
requestcount	Number	CountStatistic	The total number of requests processed so far.

Statistics available for web modules are shown in [“Web Container Statistics” on page 200](#).

TABLE 16-8 Web Container (Web Module) Statistics

Statistic	Data Type	Comments
jspcount	CountStatistic	Number of JSP pages that have been loaded in the web module.
jspreloadcount	CountStatistic	Number of JSP pages that have been reloaded in the web module.
sessiontotal	CountStatistic	Total number of sessions that have been created for the web module.
activesessionscurrent	CountStatistic	Number of currently active sessions for the web module.
activesessionshigh	CountStatistic	Maximum number of concurrently active sessions for the web module.
rejectedsessiontotal	CountStatistic	Total number of rejected sessions for the web module. This is the number of sessions that were not created because the maximum allowed number of sessions were active.
expiredsessiontotal	CountStatistic	Total number of expired sessions for the web module.
sessionsize (EE only)	AverageRangeStatistic	Size of the session for the web module. Value is either high, low, or average, or is in bytes for serialized sessions.

TABLE 16–8 Web Container (Web Module) Statistics (Continued)

Statistic	Data Type	Comments
containerlatency (EE only)	AverageRangeStatistic	Latency for the web container’s part of the overall latency request. Value is either high, low, or average.
sessionpersisttime (EE only)	AverageRangeStatistic	Time (in ms, low, high, or average) taken to persist HTTP session state to back-end store for the web module.
cachedsessionscurrent (EE only)	CountStatistic	Current number of sessions cached in memory for the web module.
passivatedsessionscurrent (EE only)	CountStatistic	Current number of sessions passivated for the web module.

HTTP Service Statistics

The statistics available for the HTTP service are shown in [“HTTP Service Statistics” on page 202](#). These statistics are applicable to the Platform Edition only.

TABLE 16–9 HTTP Service Statistics (applicable to Platform Edition only)

Statistic	Units	Data Type	Comments
bytesreceived	Bytes	CountStatistic	The cumulative value of the bytes received by each of the request processors.
bytessent	Bytes	CountStatistic	The cumulative value of the bytes sent by each of the request processors.
currentthreadcount	Number	CountStatistic	The number of processing threads currently in the listener thread pool.
currentthreadsbusy	Number	CountStatistic	The number of request processing threads currently in use in the listener thread pool serving requests.
errorcount	Number	CountStatistic	The cumulative value of the error count, which represents the number of cases where the response code is greater than or equal to 400.
maxsparethreads	Number	CountStatistic	The maximum number of unused response processing threads that can exist.

TABLE 16–9 HTTP Service Statistics (applicable to Platform Edition only) *(Continued)*

Statistic	Units	Data Type	Comments
minsparethreads	Number	CountStatistic	The minimum number of unused response processing threads that can exist.
maxthreads	Number	CountStatistic	The maximum number of request processing threads created by the listener.
maxtime	Milliseconds	CountStatistic	The maximum amount of time for processing threads.
processing-time	Milliseconds	CountStatistic	The cumulative value of the times taken to process each request. The processing time is the average of request processing times divided by the request count.
request-count	Number	CountStatistic	The total number of requests processed so far.

JDBC Connection Pools Statistics

Monitor JDBC resources to measure performance and capture resource usage at runtime. As the creation of JDBC connections are expensive and frequently cause performance bottlenecks in applications, it is crucial to monitor how a JDBC connection pool is releasing and creating new connections and how many threads are waiting to retrieve a connection from a particular pool.

The statistics available for the JDBC connection pool are shown in the following table.

TABLE 16–10 JDBC Connection Pool Statistics

Statistic	Units	Data Type	Description
numconnfailedvalidation	Number	CountStatistic	The total number of connections in the connection pool that failed validation from the start time until the last sample time.
numconnused	Number	RangeStatistic	Provides connection usage statistics. The total number of connections that are currently being used, as well as information about the maximum number of connections that were used (the high water mark).

TABLE 16–10 JDBC Connection Pool Statistics (Continued)

Statistic	Units	Data Type	Description
numconnfree	Number	RangeStatistic	The total number of free connections in the pool as of the last sampling.
numconntimedout	Number	BoundedRangeStatistic	The total number of connections in the pool that timed out between the start time and the last sample time.
averageconnwaittime	Number	CountStatistic	Indicates the average wait time of connections for successful connection request attempts to the connector connection pool.
waitqueuelength	Number	CountStatistic	Number of connection requests in the queue waiting to be serviced.
connectionrequestwaittime		RangeStatistic	The longest and shortest wait times of connection requests. The current value indicates the wait time of the last request that was serviced by the pool.
numconncreated	Milliseconds	CountStatistic	The number of physical connections that were created since the last reset.
numconndestroyed	Number	CountStatistic	Number of physical connections that were destroyed since the last reset.
numconnacquired	Number	CountStatistic	Number of logical connections acquired from the pool.
numconnreleased	Number	CountStatistic	Number of logical connections released to the pool.

JMS/Connector Service Statistics

The statistics available for the connector connection pools are shown in “[JMS/Connector Service Statistics](#)” on page 204. Statistics for Connector Work Management are shown in “[JMS/Connector Service Statistics](#)” on page 204.

TABLE 16–11 Connector Connection Pool Statistics

Statistic	Units	Data Type	Description
numconnfailedvalidation	Number	CountStatistic	The total number of connections in the connection pool that failed validation from the start time until the last sample time.
numconnused	Number	RangeStatistic	Provides connection usage statistics. The total number of connections that are currently being used, as well as information about the maximum number of connections that were used (the high water mark).
numconnfree	Number	RangeStatistic	The total number of free connections in the pool as of the last sampling.
numconn timedout	Number	CountStatistic	The total number of connections in the pool that timed out between the start time and the last sample time.
averageconnwaittime	Number	CountStatistic	Average wait time of connections before they are serviced by the connection pool.
waitqueue length	Number	CountStatistic	Number of connection requests in the queue waiting to be serviced.
connectionrequestwaittime		RangeStatistic	The longest and shortest wait times of connection requests. The current value indicates the wait time of the last request that was serviced by the pool.
numconncreated	Milliseconds	CountStatistic	The number of physical connections that were created since the last reset.
numconndestroyed	Number	CountStatistic	Number of physical connections that were destroyed since the last reset.
numconnacquired	Number	CountStatistic	Number of logical connections acquired from the pool.
numconnreleased	Number	CountStatistic	Number of logical connections released to the pool.

Statistics available for Connector Work Management are listed in [“JMS/Connector Service Statistics” on page 204](#).

TABLE 16–12 Connector Work Management Statistics

Statistic	Data Type	Description
activeworkcount	RangeStatistic	Number of work objects executed by the connector.
waitqueuelength	RangeStatistic	Number of work objects waiting in the queue before executing.
workrequestwaittime	RangeStatistic	Longest and shortest wait of a work object before it gets executed.
submittedworkcount	CountStatistic	Number of work objects submitted by a connector module.
rejectedworkcount	CountStatistic	Number of work objects rejected by the Application Server.
completedworkcount	CountStatistic	Number of work objects that were completed.

Statistics for Connection Managers in an ORB

The statistics available for the connection manager in an ORB are listed in [“Statistics for Connection Managers in an ORB” on page 206](#).

TABLE 16–13 Connection Manager (in an ORB) Statistics

Statistic	Units	Data Type	Description
connectionsidle	Number	CountStatistic	Provides total number of connections that are idle to the ORB.
connectionsinuse	Number	CountStatistic	Provides total number of connections in use to the ORB.
totalconnections	Number	BoundedRangeStatistic	Total number of connections to the ORB.

Thread Pools Statistics

The statistics available for the thread pool are shown in the following table.

TABLE 16–14 Thread Pool Statistics

Statistic	Units	Data Type	Description
averagetimeinqueue	Milliseconds	RangeStatistic	The average amount of time in milliseconds a request waited in the queue before getting processed.

TABLE 16-14 Thread Pool Statistics *(Continued)*

Statistic	Units	Data Type	Description
averageworkcompletion-time	Milliseconds	RangeStatistic	The average amount of time taken to complete an assignment, in milliseconds.
currentnumberofthreads	Number	BoundedRangeStatistic	Current number of request processing threads.
numberofavailablethreads	Number	CountStatistic	The number of threads that are available.
numberofbusythreads	Number	CountStatistic	The number of threads that are busy.
totalworkitemsadded	Number	CountStatistic	The total number of work items added so far to the work queue.

Transaction Service Statistics

The transaction service allows the client to freeze the transaction subsystem in order to roll back transactions and determine the transactions that are in process at the time of the freeze. The statistics available for the transaction service are shown in the following table.

TABLE 16-15 Transaction Service Statistics

Statistic	Data Type	Description
activecount	CountStatistic	Number of transactions currently active.
activeids	StringStatistic	The ID's of the transactions that are currently active. Every such transaction can be rolled back after freezing the transaction service.
committedcount	CountStatistic	Number of transactions that have been committed.
rolledbackcount	CountStatistic	Number of transactions that have been rolled back.
state	StringStatistic	Indicates whether or not the transaction has been frozen.

Java Virtual Machine (JVM) Statistics

The JVM has monitorable attributes that are always enabled. The statistics available for the JVM are shown in the following table.

TABLE 16–16 JVM Statistics

Statistic	Data Type	Description
heapsize	BoundedRangeStatistic	The resident memory footprint with the higher and lower bounds of the JVM's memory heap size.
uptime	CountStatistic	The amount of time the JVM has been running.

JVM Statistics in J2SE 5.0

If the Application Server is configured to run on J2SE version 5.0 or higher, additional monitoring information can be obtained from the JVM. Set the monitoring level to LOW to enable the display of this additional information. Set the monitoring level to HIGH to also view information pertaining to each live thread in the system. More information on the additional monitoring features available in J2SE 5.0 is available in a document titled *Monitoring and Management for the Java Platform*, which is available from <http://java.sun.com/j2se/1.5.0/docs/guide/management/>.

The J2SE 5.0 monitoring tools are discussed at <http://java.sun.com/j2se/1.5.0/docs/tooldocs/#manage>.

The statistics available for class loading in the JVM in J2SE 5.0 are shown in “JVM Statistics in J2SE 5.0” on page 208.

TABLE 16–17 JVM Statistics for J2SE 5.0 - Class Loading

Statistic	Data Type	Description
loadedclasscount	CountStatistic	Number of classes that are currently loaded in the JVM.
totalloadedclasscount	CountStatistic	Total number of classes that have been loaded since the JVM began execution.
unloadedclasscount	CountStatistic	Number of classes that have been unloaded from the JVM since the JVM began execution.

The statistics available for compilation in the JVM in J2SE 5.0 are shown in “JVM Statistics in J2SE 5.0” on page 208.

TABLE 16–18 JVM Statistics for J2SE 5.0 - Compilation

Statistic	Data Type	Description
totalcompilationtime	CountStatistic	Accumulated time (in milliseconds) spent in compilation.

The statistics available for garbage collection in the JVM in J2SE 5.0 are shown in [“JVM Statistics in J2SE 5.0” on page 208](#).

TABLE 16–19 JVM Statistics for J2SE 5.0 - Garbage Collection

Statistic	Data Type	Description
collectioncount	CountStatistic	Total number of collections that have occurred.
collectiontime	CountStatistic	Accumulated collection time (in milliseconds).

The statistics available for memory in the JVM in J2SE 5.0 are shown in [“JVM Statistics in J2SE 5.0” on page 208](#).

TABLE 16–20 JVM Statistics for J2SE 5.0 - Memory

Statistic	Data Type	Description
objectpendingfinalizationcount	CountStatistic	Approximate number of objects that are pending finalization.
initheapsize	CountStatistic	Size of the heap initially requested by the JVM.
usedheapsize	CountStatistic	Size of the heap currently in use.
maxheapsize	CountStatistic	Maximum amount of memory (in bytes) that can be used for memory management.
committedheapsize	CountStatistic	Amount of memory (in bytes) that is committed for the JVM to use.
initnonheapsize	CountStatistic	Size of the non-heap area initially requested by the JVM.
usednonheapsize	CountStatistic	Size of the non-heap area currently in use.
maxnonheapsize	CountStatistic	Maximum amount of memory (in bytes) that can be used for memory management.

TABLE 16–20 JVM Statistics for J2SE 5.0 - Memory (Continued)

Statistic	Data Type	Description
committednonheapsize	CountStatistic	Amount of memory (in bytes) that is committed for the JVM to use.

The statistics available for the operating system in the JVM in J2SE 5.0 are shown in [“JVM Statistics in J2SE 5.0” on page 208](#).

TABLE 16–21 JVM Statistics for J2SE 5.0 - Operating System

Statistic	Data Type	Description
arch	StringStatistic	Operating system architecture.
availableprocessors	CountStatistic	Number of processors available to the JVM.
name	StringStatistic	Operating system name.
version	StringStatistic	Operating system version.

The statistics available for the runtime in the JVM in J2SE 5.0 are shown in [“JVM Statistics in J2SE 5.0” on page 208](#).

TABLE 16–22 JVM Statistics for J2SE 5.0 - Runtime

Statistic	Data Type	Description
name	StringStatistic	Name representing the running JVM
vmname	StringStatistic	JVM implementation name.
vmvendor	StringStatistic	JVM implementation vendor.
vmversion	StringStatistic	JVM implementation version.
specname	StringStatistic	JVM specification name.
specvendor	StringStatistic	JVM specification vendor.
specversion	StringStatistic	JVM specification version.
managementspecversion	StringStatistic	Management spec. version implemented by the JVM.
classpath	StringStatistic	Classpath that is used by the system class loader to search for class files.
librarypath	StringStatistic	Java library path.

TABLE 16–22 JVM Statistics for J2SE 5.0 - Runtime (Continued)

Statistic	Data Type	Description
bootclasspath	StringStatistic	Classpath that is used by the bootstrap class loader to search for class files.
inputarguments	StringStatistic	Input arguments passed to the JVM. Does not include the arguments to the main method.
uptime	CountStatistic	Uptime of the JVM (in milliseconds).

The statistics available for ThreadInfo in the JVM in J2SE 5.0 are shown in [“JVM Statistics in J2SE 5.0” on page 208](#).

TABLE 16–23 JVM Statistics for J2SE 5.0 - Thread Info

Statistic	Data Type	Description
threadid	CountStatistic	Id of the thread.
threadname	StringStatistic	Name of the thread.
threadstate	StringStatistic	State of the thread.
blockedtime	CountStatistic	Time elapsed (in milliseconds) since the thread entered the BLOCKED state. Returns -1 if thread contention monitoring is disabled.
blockedcount	CountStatistic	Total number of times that the thread entered the BLOCKED state.
waitedtime	CountStatistic	Elapsed time (in milliseconds) that the thread has been in a WAITING state. Returns -1 if thread contention monitoring is disabled.
waitedcount	CountStatistic	Total number of times the thread was in WAITING or TIMED_WAITING states.
lockname	StringStatistic	String representation of the monitor lock that the thread is blocked to enter or waiting to be notified through the Object.wait method.
lockownerid	CountStatistic	Id of the thread that holds the monitor lock of an object on which this thread is blocking.

TABLE 16–23 JVM Statistics for J2SE 5.0 - Thread Info (Continued)

Statistic	Data Type	Description
lockownername	StringStatistic	Name of the thread that holds the monitor lock of the object this thread is blocking on.
stacktrace	StringStatistic	Stack trace associated with this thread.

The statistics available for threads in the JVM in J2SE 5.0 are shown in “[JVM Statistics in J2SE 5.0](#)” on page 208.

TABLE 16–24 JVM Statistics for J2SE 5.0 - Threads

Statistic	Data Type	Description
threadcount	CountStatistic	Current number of live daemon and non-daemon threads.
peakthreadcount	CountStatistic	Peak live thread count since the JVM started or the peak was reset.
totalstartedthreadcount	CountStatistic	Total number of threads created and/or started since the JVM started.
daemonthreadcount	CountStatistic	Current number of live daemon threads.
allthreadids	StringStatistic	List of all live thread ids.
currentthreadcputime	CountStatistic	CPU time for the current thread (in nanoseconds) if CPU time measurement is enabled. If CPU time measurement is disabled, returns -1.
monitordeadlockedthreads	StringStatistic	List of thread ids that are monitor deadlocked.

Enabling and Disabling Monitoring

- “[To configure monitoring levels using the Admin Console](#)” on page 212
- “[To configure monitoring using the asadmin tool](#)” on page 213

To configure monitoring levels using the Admin Console

To configure monitoring in the Admin Console, go to Application Server node > Configuration > Monitoring

By default, monitoring is turned off for all components and services. To turn monitoring on, select LOW or HIGH from the combo box. To turn monitoring off, select OFF from the combo box.

For details on configuring monitoring, see the online help available with the Admin Console.

Equivalent asadmin commands are `get` and `set`.

For example, to turn on monitoring for the HTTP Service, use the following `asadmin` command:

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=LOW
```

▼ To configure monitoring using the asadmin tool

Use the `asadmin` `get` and `set` commands to configure monitoring from the command line.

1 Use the `get` command to find out what services and components currently have monitoring enabled.

```
asadmin> get --user admin-user server.monitoring-service.module-monitoring-levels.*
```

Returns:

```
server.monitoring-service.module-monitoring-levels.
connector-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.
connector-service = OFF
server.monitoring-service.module-monitoring-levels.ejb-container = OFF
server.monitoring-service.module-monitoring-levels.http-service = OFF
server.monitoring-service.module-monitoring-levels.jdbc-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.jms-service = OFF
server.monitoring-service.module-monitoring-levels.jvm = OFF
server.monitoring-service.module-monitoring-levels.orb = OFF
server.monitoring-service.module-monitoring-levels.thread-pool = OFF
server.monitoring-service.module-monitoring-levels.transaction-service = OFF
server.monitoring-service.module-monitoring-levels.web-container = OFF
```

2 Use the `set` command to enable monitoring.

For example, to enable monitoring for the HTTP service:

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=LOW
```

To disable monitoring, use the `set` command and specify `OFF` for the monitoring level.

Viewing Monitoring Data

- [“Viewing monitoring data in the Admin Console” on page 214](#)
- [“Viewing Monitoring Data With the asadmin Tool” on page 214](#)

Viewing monitoring data in the Admin Console

To view monitoring data in the Admin Console, go to Application Server node > Monitor, and select a component or service that has been deployed onto the server instance and for which monitoring is enabled.

On this page, it is possible to select and view monitoring data for JVM, server, applications, thread pools, HTTP service, transaction service, log statistics, and call flow statistics. A diagram showing how these components and services are organized is shown in [“About the Tree Structure of Monitorable Objects” on page 192](#).

For details on viewing or configuring monitoring, see the online help available with the Admin Console.

For more information on the attributes for each component or service, refer to [“About Statistics for Monitored Components and Services” on page 196](#).

The equivalent asadmin command is `get --monitor`

For example, to view monitoring data for the JVM, use the following asadmin command:

```
asadmin> get --user adminuser --monitor server.jvm.*
```

Viewing Monitoring Data With the asadmin Tool

- [“To use the asadmin tool to view monitoring data” on page 214](#)
- [“Understanding and Specifying Dotted Names” on page 216](#)
- [“Examples of the list and get Commands” on page 217](#)
- [“Examples for the list --user admin-user --monitor Command” on page 217](#)
- [“Examples for the get --user admin-user --monitor Command” on page 218](#)
- [“To use the PetStore example” on page 219](#)
- [“Expected Output for list and get Commands at All Levels” on page 223](#)

▼ To use the asadmin tool to view monitoring data

To view monitoring data using the asadmin tool, use the `asadmin list` and `asadmin get` commands followed by the dotted name of a monitorable object, as described in the following section.

1 To view the names of the objects that can be monitored, use the `asadmin list` command.

For example, to view a list of application components and subsystems that have monitoring enabled for the server instance, type the following command in a terminal window:

```
asadmin> list --user adminuser --monitor server
```

The preceding command returns a list of application components and subsystems that have monitoring enabled, for example:

```
server.resources
server.connector-service
server.orb
server.jms-service
server.jvm
server.applications
server.http-service
server.thread-pools
```

For further information on using the `list` command, refer to [list\(1\)](#).

For further examples using the `list` command, refer to “[Examples of the list and get Commands](#)” on [page 217](#). For further information on the dotted names you can use with the `list` command, refer to “[Understanding and Specifying Dotted Names](#)” on [page 216](#).

2 To display monitoring statistics for an application component or subsystem for which monitoring has been enabled, use the `asadmin get` command.

To get the statistics, type the `asadmin get` command in a terminal window, specifying a name displayed by the `list` command in the preceding step. The following example attempts to get all attributes from a subsystem for a specific object:

```
asadmin> get --user adminuser --monitor server.jvm.*
```

The command returns the following attributes and data:

```
server.jvm.dotted-name = server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information about
    the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
    been running.
```

```
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

For further information using the `get` command, refer to [get\(1\)](#).

For further examples using the `get` command, refer to “[Examples of the list and get Commands](#)” on page 217. For further information on the dotted names you can use with the `get` command, refer to “[Understanding and Specifying Dotted Names](#)” on page 216.

Understanding and Specifying Dotted Names

In the `asadmin list` and `get` commands, specify the dotted name of monitorable objects. All child objects are addressed using the dot (.) character as separator, thus these are referred to as *dotted names*. 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-service` is one of the valid monitorable object types and is a singleton. To address a singleton child node representing the `http-service` of instance `server`, the dotted name is:

```
server.http-service
```

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

```
server.applications.petstore
```

The dotted names can also address specific attributes in monitorable objects. For example, `http-service` has a monitorable attribute called `bytesreceived-lastsampletime`. The following name addresses the `bytesreceived` attribute:

```
server.http-service.server.http-listener-1.
  bytesreceived-lastsampletime
```

The administrator is not expected to know the valid dotted names for `asadmin list` and `get` commands. The `list` command displays available monitorable objects, while the `get` command used with a wildcard parameter allows the inspection of all available attributes on any monitorable object.

The underlying assumptions for using the `list` and `get` commands with dotted names are:

- Any `list` command that has a dotted name that is **not** followed by a wildcard (*) gets as its result the current node’s immediate children. For example, `list --user adminuser --monitor server` lists all immediate children belonging to the `server` node.

- Any `list` command that has a dotted name followed by a wildcard of the form `. *` gets as its result a hierarchical tree of children nodes from the current node. For example, `list --user adminuser --monitor server.applications.*` lists all children of `applications` and their subsequent child nodes and so on.
- Any `list` command that has a dotted name preceded or followed by a wildcard of the form `*dottedname` or `dotted * name` or **dotted name *** gets as its result all nodes and their children matching the regular expression created by the provided matching pattern.
- A `get` command followed by a `. *` or a `* *` gets as its result the set of attributes and their values belonging to the current node to be matched.

For more information, read [“Expected Output for list and get Commands at All Levels” on page 223](#).

Examples of the list and get Commands

This section contains the following topics:

- [“Examples for the list --user admin-user --monitor Command” on page 217](#)
- [“Examples for the get --user admin-user --monitor Command” on page 218](#)

Examples for the list --user admin-user --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 subcomponents for a server instance. For a more complete listing of list examples, see [“Expected Output for list and get Commands at All Levels” on page 223](#).

Example 1

```
asadmin> list --user admin-user --monitor server
```

The preceding command returns a list of application components and subsystems that have monitoring enabled, for example:

```
server.resources
server.orb
server.jvm
server.jms-service
server.connector-service
server.applications
server.http-service
server.thread-pools
```

It is also possible to list applications that are currently monitored in the specified server instance. This is useful when particular monitoring statistics are sought from an application using the `get` command.

Example 2

```
asadmin> list --user admin-user --monitor server.applications
```

Returns:

```
server.applications.adminapp
  server.applications.admingui
server.applications.myApp
```

For a more comprehensive example, see [“To use the PetStore example” on page 219](#).

Examples for the get --user admin-user --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 [“Expected Output for list and get Commands at All Levels” on page 223](#) for more information on the use of the get command.

Example 1

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

```
asadmin> get --user admin-user --monitor server.jvm.*
```

Returns:

```
server.jvm.dotted-name= server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information about
  the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
  been running.
```

```
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

Example 2

Attempt to get all attributes from a J2EE application:

```
asadmin> get --user admin-user --monitor server.applications.myJ2eeApp.*
```

Returns:

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

There are no monitorable attributes exposed at the J2EE-application level, therefore this reply displays.

Example 3

Attempt to get a specific attribute from a subsystem:

```
asadmin> get --user admin-user --monitor server.jvm.uptime-lastsampletime
```

Returns:

```
server.jvm.uptime-lastsampletime = 1093215374813
```

Example 4

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

```
asadmin> get --user admin-user --monitor server.jvm.badname
```

Returns:

```
No such attribute found from reflecting the corresponding Stats
interface: [badname]
CLI137 Command get failed.
```

▼ To use the PetStore example

The following example illustrates how the `asadmin` tool might be used for monitoring purposes.

A user wants to inspect the number of calls made to a method in the sample `PetStore` application after it has been deployed onto the Application Server. The instance onto which it has been deployed is named `server`. A combination of the `list` and `get` commands are used to access desired statistics on a method.

1 Start the Application Server and the asadmin tool.**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=4849
```

3 List monitorable components for instance server:

```
asadmin> list --user adminuser --monitor server*
```

Returns (output will be similar to):

```
server
server.applications
server.applications.CometEJB
server.applications.ConverterApp
server.applications.petstore
server.http-service
server.resources
server.thread-pools
```

The list of monitorable components includes thread-pools, http-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 server.applications.petstore
```

Returns:

```
server.applications.petstore.signon-ejb_jar
server.applications.petstore.catalog-ejb_jar
server.applications.petstore.uidgen-ejb_jar
server.applications.petstore.customer-ejb_jar
server.applications.petstore.petstore-ejb_jar
server.applications.petstore.petstore\war
server.applications.petstore.AsyncSenderJAR_jar
server.applications.petstore.cart-ejb_jar
```

5 List the monitorable subcomponents in the EJB module signon-ejb_jar of the PetStore application:

```
asadmin> list -m server.applications.petstore.signon-ejb_jar
```

Returns:

```
server.applications.petstore.signon-ejb_jar.SignOnEJB
server.applications.petstore.signon-ejb_jar.UserEJB
```

6 List the monitorable subcomponents in the entity bean UserEJB for the EJB module signon-ejb_jar of the PetStore application:

```
asadmin> list -m server.applications.petstore.signon-ejb_jar.UserEJB
```

Returns (with dotted name removed for space considerations):

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-cache
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods
server.applications.petstore.signon-ejb_jar.UserEJB.bean-pool
```

7 List the monitorable subcomponents in the method getUsername for the entity bean UserEJB in the EJB module signon-ejb_jar of the PetStore application:

```
asadmin> list -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUsername
```

Returns:

Nothing to list at server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUsername. To get the valid names beginning with a string, use the wildcard "*" character. For example, to list all names that begin with "server", use "list server*".

8 There are no monitorable subcomponents for methods. Get all monitorable statistics for the method getUsername.

```
asadmin> get -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUsername.*
```

Returns:

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUsername.executiontime-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUsername.executiontime-description = Provides the time in milliseconds
    spent during the last successful/unsuccessful attempt to execute the
    operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUsername.executiontime-lastsampletime = 1079981809259
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUsername.executiontime-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUsername.executiontime-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUsername.executiontime-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUsername.methodstatistic-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUsername.methodstatistic-description = Provides the number of times an
    operation was called, the total time that was spent during the
```

```
invocation and so on.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-lastsamptime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-maxtime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-mintime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-totaltime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-unit =
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-description = Provides the total number of errors
that occurred during invocation or execution of an operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-lastsamptime = 1079981809273
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-name = TotalNumErrors
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-description = Provides the total number of
successful invocations of the method.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-lastsamptime = 1079981809255
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-name = TotalNumSuccess
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-unit = count
```

9 To also get a specific statistic, such as execution time, use a command such as the following:

```
asadmin> get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.executiontime-count
```

Returns:

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 1
```

Expected Output for list and get Commands at All Levels

The following tables show the command, dotted name, and corresponding output at each level of the tree.

TABLE 16–25 Top Level

Command	Dotted Name	Output
list -m	server	server.applicationsserver.thread-poolsserver. resourceesserver.http-serviceserver.transaction- serviceserver.orb.connection-managersserver.orb. connection-managers.orb\Connections\Inbound\ AcceptedConnectionsserver.jvm
list -m	server.*	Hierarchy of child nodes below this node.
get -m	server.*	No output except a message saying there are no attributes at this node.

The following table shows the command, dotted name, and corresponding output for the applications level.

TABLE 16–26 Applications Level

Command	Dotted Name	Output
list -m	server.applications or *applications	appllapp2web-module1_warejb-module2_jar...
list -m	server.applications.* or *applications.*	Hierarchy of child nodes below this node.
get -m	server.applications.* or *applications.*	No output except message saying there are no attributes at this node.

The following table shows the command, dotted name, and corresponding output for stand-alone modules and enterprise applications at the applications level.

TABLE 16-27 Applications - Enterprise Applications and Standalone Modules

Command	Dotted Name	Output
list -m	server.applications.app1 or *app1 Note: this level is only applicable if an enterprise application has been deployed. It is not applicable if a standalone module is deployed.	ejb-module1_jarweb-module2_warejb-module3_jarweb-module3_war...
list -m	server.applications.app1.* or *app1.*	Hierarchy of child nodes below this node.
get -m	server.applications.app1.* or *app1.*	No output except message saying there are no attributes at this node.
list -m	server.applications.app1.ejb-module1_jar or *ejb-module1_jar or server.applications.ejb-module1_jar	bean1bean2bean3...
list -m	server.applications.app1.ejb-module1_jar or *ejb-module1_jar or server.applications.ejb-module1_jar	Hierarchy of child nodes below this node.
get -m	server.applications.app1.ejb-module1_jar.* or *ejb-module1_jar.* or server.applications.ejb-module1_jar.*	No output except message saying there are no attributes at this node.

TABLE 16-27 Applications - Enterprise Applications and Standalone Modules (Continued)

Command	Dotted Name	Output
list -m	server.applications.app1. ejb-module1_jar.bean1 Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.	List of child nodes: bean-poolbean-cachebean-method
list -m	server.applications.app1. ejb-module1_jar.bean1 Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.	Hierarchy of child nodes and a list of all attributes for this node and for any subsequent child nodes.
get -m	server.applications.app1. ejb-module1_jar.bean1.* Note: In standalone modules, the node containing the application name (app1 in this example) does not appear.	The following attributes and their associated values: CreateCount_CountCreateCount_ DescriptionCreateCount_ LastSampleTimeCreateCount_ NameCreateCount_ StartTimeCreateCount_ UnitMethodReadyCount_ CurrentMethodReadyCount_ DescriptionMethodReadyCount_ HighWaterMarkMethodReadyCount_ LastSampleTimeMethodReadyCount_ LowWaterMarkMethodReadyCount_ NameMethodReadyCount_ StartTimeMethodReadyCount_ UnitRemoveCount_CountRemoveCount_ DescriptionRemoveCount_ LastSampleTimeRemoveCount_ NameRemoveCount_StartTimeAttribute RemoveCount_Unit
list -m	server.applications.app1. ejb-module1_jar.bean1.bean-pool Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.	No attributes, but a message saying: Nothing to list at server.applications.app1. ejb-module1_jar.bean1-cache. To get the valid names beginning with a string, use the wildcard (*) character. For example, to list all names that begin with server, use list server*.
get -m	server.applications.app1. ejb-module1_jar.bean1.bean-pool.* Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.	List of attributes and values corresponding to EJB Pool attributes as described in Table 1-4.

TABLE 16–27 Applications - Enterprise Applications and Standalone Modules *(Continued)*

Command	Dotted Name	Output
list -m	server.applications.app1. ejb-module1_jar.bean1.bean-cache Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.	No attributes, but a message saying “Use get command with the --monitor option to view this node’s attributes and values.”
get -m	server.applications.app1. ejb-module1_jar.bean1.bean-cache.* Note: In standalone modules, the node containing the application name (app1 in this example) does not appear.	List of attributes and values corresponding to EJB Cache attributes as described in Table 1-5.
list -m	server.applications.app1. ejb-module1_jar.bean1.bean-method.method1 Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.	No attributes, but a message saying “Use get command with the --monitor option to view this node’s attributes and values.”
get -m	server.applications.app1. ejb-module1_jar.bean1.bean-method.method1.* Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.	List of attributes and values corresponding to EJB Methods attributes as described in Table 1-2.
list -m	server.applications.app1.web-module1_war	Displays the virtual server(s) assigned to the module.
get -m	server.applications.app1.web-module1_war.*	No output except a message saying there are no attributes at this node.
list -m	server.applications.app1.web-module1_war. virtual_server	Displays list of servlets registered.
get -m	server.applications.app1.web-module1_war. virtual_server.*	No output except a message saying there are no attributes at this node.
list -m	server.applications.app1.web-module1_war. virtual_server.servlet1	No attributes, but a message saying “Use get command with the --monitor option to view this node’s attributes and values.”
get -m	server.applications.app1.web-module1_war. virtual_server.servlet1.*	List of attributes and values corresponding to web container (Servlet) attributes as described in Table 1-7.

The following table shows the command, dotted name, and corresponding output for the HTTP Service level.

TABLE 16–28 HTTP-Service Level

Command	Dotted Name	Output
list -m	server.http-service	List of virtual servers.
get -m	server.http-service.*	No output except message saying there are no attributes at this node.
list -m	server.http-service.server	List of HTTP Listeners.
get -m	server.http-service.server.*	No output except message saying there are no attributes at this node.
list -m	server.http-service.server.http-listener1	No attributes, but a message saying “Use get command with the --monitor option to view this node’s attributes and values.”
get -m	server.http-service.server.*	List of attributes and values corresponding to HTTP Service attributes as described in Table 1-9.

The following table shows the command, dotted name, and corresponding output for the thread pools level.

TABLE 16–29 Thread-Pools Level

Command	Dotted Name	Output
list -m	server.thread-pools	List of thread-pool names.
get -m	server.thread-pools.*	No output except message saying there are no attributes at this node.
list -m	server.thread-pools.orb\threadpool\thread-pool-1	No attributes, but a message saying “Use get command with the --monitor option to view this node’s attributes and values.”
get -m	server.thread-pools.orb\threadpool\thread-pool-1.*	List of attributes and values corresponding to Thread Pool attributes as described in Table 1-14.

The following table shows the command, dotted name, and corresponding output for the resources level.

TABLE 16–30 Resources Level

Command	Dotted Name	Output
list -m	server.resources	List of pool names.
get -m	server.resources.*	No output except message saying there are no attributes at this node.
list -m	server.resources.jdbc-connection-pool-pool.connection-pool1	No attributes, but a message saying “Use get command with the --monitor option to view this node’s attributes and values.”
get -m	server.resources.jdbc-connection-pool-pool.connection-pool1.*	List of attributes and values corresponding to Connection Pool attributes as described in Table 1-10.

The following table shows the command, dotted name, and corresponding output for the transaction service level.

TABLE 16–31 Transaction-Service Level

Command	Dotted Name	Output
list -m	server.transaction-service	No attributes, but a message saying “Use get command with the --monitor option to view this node’s attributes and values.”
get -m	server.transaction-service.*	List of attributes and values corresponding to Transaction Service attributes as described in Table 1-15.

The following table shows the command, dotted name, and corresponding output for the ORB level.

TABLE 16–32 ORB Level

Command	Dotted Name	Output
list -m	server.orb	server-orb.connection-managers
get -m	server.orb.*	No output except message saying there are no attributes at this node.
list -m	server.orb.connection-managers	Name(s) of ORB connection managers.

TABLE 16–32 ORB Level (Continued)

Command	Dotted Name	Output
get -m	server.orb.connection-managers.*	No output except message saying there are no attributes at this node.
list -m	server.orb.connection-managers. orb\Connections\Inbound\ .AcceptedConnections	No attributes, but a message saying “Use get command with the --monitor option to view this node’s attributes and values.”
get -m	server.orb.connection-managers. orb\Connections\Inbound\ .AcceptedConnections.*	List of attributes and values corresponding to ORB Connection Manager attributes as described in Table 1-13.

The following table shows the command, dotted name, and corresponding output for the JVM level.

TABLE 16–33 JVM Level

Command	Dotted Name	Output
list -m	server.jvm	No attributes, but a message saying “Use get command with the --monitor option to view this node’s attributes and values.”
get -m	server.jvm.*	List of attributes and values corresponding to JVM attributes as described in Table 1-16.

Using JConsole

This section contains the following topics:

- [“Securing JConsole to Application Server Connection” on page 230](#)
- [“Prerequisites for Connecting JConsole to Application Server” on page 231](#)
- [“Connecting JConsole to Application Server” on page 231](#)
- [“Connecting JConsole Securely to Application Server” on page 232](#)

Administration (management and monitoring) of the Application Server is based on JMX. This means that the managed components are represented as MBeans. With Java 2 Standard Edition (J2SE) 5.0, you can monitor the JVM and view the JVM MBeans to understand what is going on there. To expose this instrumentation, Application Server provides a configuration of Standard JMX Connector Server called System JMX Connector Server. When the Application Server starts up, an instance of this JMX Connector Server starts up, exposing the instrumentation to trusted clients.

The Java Monitoring and Management Console (JConsole) is a popular JMX Connector that can manage a JMX backend. JConsole (<http://java.sun.com/j2se/1.5.0/docs/tooldocs/share/jconsole.html>) is available as part of the standard JDK distribution beginning with J2SE 5.0. For more information on JConsole, see <http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html>

When you configure JConsole with Application Server, the Application Server becomes the JMX Connector's server-end and JConsole becomes the JMX Connector's preferred client-end. “Connecting JConsole to Application Server” on page 231 shows how to make a successful connection .

Securing JConsole to Application Server Connection

There are subtle differences in how to connect to Application Server, or any JMX Connector Server end based on the transport layer security of the connection. If the server end is secure (guarantees transport layer security), there is a little more configuration to be performed on the client end.

- By default, Platform Edition of Application Server has System JMX Connector Server end as insecure.
- By default, Enterprise Edition of Application Server has System JMX Connector Server end as secure.
- The protocol used for communication is RMI/JRMP. If security is enabled for the JMX Connector, the protocol used is RMI/JRMP over SSL. For details on making the JMX Connector secure, see “Configuring Security For The Admin Service’s JMX Connector” on page 119.

Note – RMI over SSL does not provide additional checks to ensure that the client is talking to the intended server. Thus, there is always a possibility, while using JConsole, that you are sending the user name and password to a malicious host. It is completely up to the administrator to make sure that security is not compromised.

When you install a Platform Edition domain on a machine such as `appserver.sun.com`, you will see the following in the DAS's (Domain Administration Server, the admin server or simply the domain) `domain.xml`:

```
<!-- The JSR 160 "system-jmx-connector" -->
<jmx-connector accept-all="false" address="0.0.0.0"
auth-realm-name="admin-realm" enabled="true" name="system" port="8686"
protocol="rmi_jrmp" security-enabled="false"/>
<!-- The JSR 160 "system-jmx-connector" -->
```

The security-enabled flag for the JMX Connector is *false*. If you are running the Enterprise Edition, or if you have turned on security for the JMX Connector in the Platform Edition, this flag is set to *true*.

```
<!-- The JSR 160 "system-jmx-connector" -->
<jmx-connector accept-all="false" address="0.0.0.0"
auth-realm-name="admin-realm" enabled="true" name="system" port="8686"
protocol="rmi_jrmp" security-enabled="true"/>
...
</jmx-connector>
<!-- The JSR 160 "system-jmx-connector" -->
```

Prerequisites for Connecting JConsole to Application Server

The JConsole setup has two parts: a server end and a client end. The Application Server domain is installed on a machine called `appserver.sun.com`, which is a powerful Solaris server. This is the server end.

The client end also has an installation of Application Server. Let us assume that the client end is a Windows machine with Java SE 5.0 and Application Server installed.

Note – The Application Server installation is needed on the client end only when your Application Server domain has security enabled on the remote machine (the default for Enterprise Edition). If you just want to administer an Application Server Platform Edition domain on the Solaris machine above, you do not need the Application Server installation on this client machine.

If the server and client ends are on the same machine, you can use `localhost` to specify the host name.

▼ Connecting JConsole to Application Server

This procedure describes connecting JConsole to Application Server without security enabled on the JMX Connector. By default, security is not enabled on Application Server Platform Edition.

- 1 **Start the domain on** `appserver.sun.com`.
- 2 **Start JConsole by running** `JDK_HOME/bin/jconsole`

- 3 **In the Connect to Agent tab of JConsole, enter user name, password, host name and port (8686, by default).**

The user name refers to the admin user name and password refers to the admin password of the domain.

- 4 **Click Connect.**

In the JConsole window you will see all your MBeans, VM information etc., in various tabs.

▼ **Connecting JConsole Securely to Application Server**

This procedure describes how to connect JConsole to Application Server with security enabled on the JMX Connector. By default, security is enabled on Application Server Enterprise Edition. Use this procedure if you have security enabled on the Platform Edition's JMX Connector.

- 1 **Install Application Server on the client machine (where JConsole is installed).**

The only reason you need this is to let JConsole know where the server certificate of the Domain Administration Server that you trust is located. To obtain that certificate, invoke at least one *remote asadmin* command and to do that, you need the local installation of Application Server.

- 2 **Start the Application Server Enterprise Edition on `appserver.sun.com`.**

Since this is an Enterprise Edition domain, the system JMX Connector server is secure. To enable security on the Platform Edition JMX Connector, see [“Configuring Security For The Admin Service’s JMX Connector” on page 119](#).

- 3 **From the local Application Server installation, run `install-dir\bin\asadmin list --user admin --secure=true --host appserver.sun.com --port 4849` (where 4849 is the server's admin port).**

Though `asadmin list` command is chosen for this example, you can run any remote `asadmin` command. You will now be prompted to accept the certificate sent by the DAS of `appserver.sun.com`.

- 4 **Press `y` to accept the certificate sent by the Domain Administration Server on `appserver.sun.com`.**

The server's certificate is stored in a file called `.asadmintruststore` in your home directory on the client machine.

Note – This step is not required if your server machine and client machine is the same. That is, if you are running JConsole also on `appserver.sun.com`.

- 5 Let JConsole know the trust store location by using the following JConsole command:**

```
JDK-dir\bin\jconsole.exe -J-Djavax.net.ssl.trustStore="C:\Documents and Settings\user\.asadmintruststore"
```

- 6 Start JConsole by running *JDK_HOME*/bin/jconsole**

- 7 In the Connect to Agent tab of JConsole, enter user name, password, host name and port (8686, by default).**

The user name refers to the admin user name and password refers to the admin password of the domain.

- 8 Click Connect.**

In the JConsole window you will see all your MBeans, VM information etc., in various tabs.

Configuring Management Rules

This section contains information about setting administration policies to automate routine administration tasks, configure self-tuning of the application server for diverse runtime condition and improve availability by preventing failures.

This section contains the following topics:

- “About Management Rules” on page 235
- “To Configure New Management Rule” on page 236
- “To Configure Monitor Properties” on page 237
- “To Configure Notification Properties” on page 238
- “To Configure Life Cycle Event Properties” on page 238
- “To Configure Log Properties” on page 239
- “To Configure Timer Event Properties” on page 239
- “To Configure Trace Event Properties” on page 239
- “Equivalent asadmin Command” on page 240

About Management Rules

Application Server administrators can use the management rules feature to set policies that self-configure and self-tune connection pool or orb thread pool, detect thread or server hangs, manage memory leaks or out of memory errors, and manage disk usage

A management rule consists of two parts — event and action. The rule specifies an action to be taken when a user—defined event occurs.

Event	An event uses the JMX notification mechanism to trigger a user-defined action.
Action	The user-defined action is performed by the custom Mbean associated with the event. The action can include restarting a server instance or increasing resources allocated to the connection pool. The action is triggered when the data

associated with the event is sent to the custom Mbean that implements the `javax.management.NotificationListener` interface.

For example, when a SEVERE message occurs in the EJB logger, the log record will be sent to the action Mbean as a field of `javax.management.LogNotification`.

The action specified in your rule is executed by an 'action' MBean. Therefore, before configuring a management rule, you should deploy a custom Mbean designed to receive event notifications and take appropriate action. For details on developing a custom Mbean and deploying it, see [Chapter 14, “Developing Custom MBeans,” in *Sun Java System Application Server Platform Edition 9 Developer’s Guide*](#).

Admin Console Tasks for Management Rules

▼ To Configure New Management Rule

To set up a new management rule, perform the following steps:

- 1 Access the Management Rules page by going to Configurations > Admin Service > Management Rules.**
- 2 Ensure All Rules is checked.**
This enables Management Rules globally. If this is unchecked none of the management rules will be executed.
- 3 Select New.**
- 4 In the Name field, provide a name for this rule. For example, Rule1.**
- 5 Check Enabled to activate this rule.**
- 6 Enter a description that identifies this rule**
- 7 Select the Event Type from the drop-down list.**

The Event Type specifies the condition to execute the rule. All events are based on the JMX notification mechanism. You can create a custom event by using Mbeans that implement `javax.management.NotificationBroadcaster/NotificationEmitter`.

The following event types are available:

- Monitor events
To monitor an attribute of an Mbean.

- Notification events
Notification events from a custom Mbean.
- Lifecycle events
Corresponds to events in the `com.sun.appserv.server.LifecycleEvent` interface.

8 Check to enable the event type to be logged.

9 Set the log level for this management rule.

The default is INFO.

10 Select Next to go to the final part of setting your management rule.

The subsequent administration screens will depend on the type of event you have selected.

▼ To Configure Monitor Properties

Provide the properties of the event type you have selected in the previous page. Monitoring properties include JSR 77 statistics.

1 Enter the Observed object name of the Mbean you want to monitor.

Specifies the `javax.management.ObjectName` of a system Mbean or the name attribute of a custom Mbean.

The `ObjectName` is of the form

domain:type=impl-class-name,name=*BeanName*,server=server, where *domain* is the JMX Domain where the MBean is registered and *BeanName* is the name of the Mbean object. For example:

user:impl-class-name=com.sun.example.mbeans.Memory,name=MemoryMBean,server=server.
You can determine the `ObjectName` of an Mbean from the deployed custom Mbean, and then add "server=server."

2 Specify an attribute of the Mbean you want to monitor.

3 Enter a description for this monitoring rule.

4 Specify the granularity, in milliseconds, at which data is collected for this attribute.

The default is 10 seconds.

5 Specify the monitor type.

Select from one from the following choices:

- Counter — If the monitor is of type `CountMonitor` or the JSR 77 statistic field being monitored is of an integral type containing a count value.

- Gauge — If the monitor is of type `GaugeMonitor` or the JSR 77 statistic field being monitored is of an integral type containing a gauge value.
 - String — If the monitor is of type `StringMonitor`
- 6 **Specify the action you want performed when the monitored data meets the condition specified in the event type.**
 - 7 **Select Finish to save your management rule.**

▼ **To Configure Notification Properties**

You can create a custom event by using Mbeans that implement `javax.management.NotificationBroadcaster/NotificationEmitter`

- 1 **Select the deployed custom Mbean from the drop-down list or specify the Object Name of the custom Mbean.**

The Object Name is of the form =
`domain:type=implementation-class-name,name=implementation-class-name`, where
domain is the JMX Domain where the MBean is registered.

- 2 **Specify the notification type.**
- 3 **Enter a description for the event associated with this management rule.**
- 4 **Specify the action you want performed when the monitored data meets the condition specified in the event type.**
- 5 **Select Finish to save your management rule.**

▼ **To Configure Life Cycle Event Properties**

Configure management rule based on Application Server's lifecycle state (ready, shutdown, termination).

- 1 **Choose the Event that will initiate the action Mbean.**

The choices are : ready, shutdown, termination

- 2 **Enter a description for the event associated with this management rule.**
- 3 **Specify the action you want performed when the monitored data meets the condition specified in the event type.**
- 4 **Select Finish to save your management rule.**

▼ To Configure Log Properties

- 1 **Select the logging interface.**
Available logging interfaces are displayed in the drop-down box.
- 2 **Select the log level for the selected logging interface.**
- 3 **Enter a description for the event associated with this management rule.**
- 4 **Specify the action you want performed when the monitored data meets the condition specified in the event type.**
- 5 **Select Finish to save your management rule.**

▼ To Configure Timer Event Properties

- 1 **Specify the date from when this rule is enabled.**
- 2 **Specify the pattern to search for.**
The default pattern is mm/dd/yyyy hh:mm:ss. The pattern must match the format in which you have specified the date.
- 3 **Specify the time period in milliseconds at which notifications will be sent to the action Mbean.**
- 4 **Specify the number of times the pattern occurs in the notification.**
- 5 **Specify the message that will be written to the log when the pattern occurs.**
- 6 **Enter a description for the event associated with this management rule.**
- 7 **Specify the action you want performed when the monitored data meets the condition specified in the event type.**
- 8 **Select Finish to save your management rule.**

▼ To Configure Trace Event Properties

- 1 **Select the trace points from the drop-down list.**
Notifications from the selected trace points are sent to the action Mbean.
- 2 **Enter a description for the event associated with this management rule.**

- 3 Specify the action you want performed when the monitored data meets the condition specified in the event type.
- 4 Select Finish to save your management rule.

Equivalent asadmin Command

To configure management rules from the command line, use the `create-management-rule` command. The command syntax is as follows:

```
create-management-rule [--terse=false] [--echo=false] [--interactive=true]
[--host localhost] [--port 4848|4849 ] [--secure | -s] --user admin_user
[--passwordfile filename] [--ruleenabled=true] [--ruledescription description]
[--action action-mbean-name] --eventtype
log|timer|trace|monitor|cluster|lifecycle|notification [--eventloglevel
FINEST|FINER|FINE|CONFIG|INFO|WARNING|SEVERE|OFF] [--recordevent=true]
[--eventdescription description] [--eventproperties
(property=value[:property=value]*)] [--target target] rule-name
```

For details on each option, see the man page for `create-management-rule` at [Sun Java System Application Server Platform Edition 9 Reference Manual](#)

Configuring the Diagnostic Service

The Diagnostic Service provides more visibility into and control of the runtime performance of a server and its applications, allowing you to diagnose and isolate faults as they occur.

This chapter contains the following sections:

- [“What is the Diagnostic Framework?” on page 241](#)
- [“Diagnostic Service Framework” on page 241](#)

What is the Diagnostic Framework?

The Application Server Diagnostic Framework is a monitoring framework that defines and implements a set of services that run within the application server standard life cycle. With the Diagnostic Service, you can define, create, collect, and access diagnostic data generated by a running server and the applications it deploys.

Diagnostic Service Framework

The diagnostic service reports configuration details for the application server instances. It is useful for diagnosing application server problems such as exceptions, performance issues, and other unexpected results. From within the Admin Console Diagnostic Service you can:

- **Compute Checksum:** Collects checksum for selective Application Server binary files under `appserver_install_dir/lib`, `appserver_install_dir/etc` and `appserver_install_dir/bin` directory
- **Verify Configuration:** Captures configuration files such as `domain.xml`, `server.policy`.
- **Capture Install Log:** Installation specific details such as Application Server version number or patch ID and contents of log file generated at the time of installation. Absolute path for installation directory is used to determine which Installer log file is collected if there are multiple installations on the same machine. The contents of `config/asenv.conf` is copied from the installation folder for DAS as well as node agents.

Installation specific details are collected only for file-based installations.

- **Capture System Information:** The following system information is collected by default:
 - Network Settings
 - OS details
 - Hardware information

Data collected using native code is not available on the Platform Edition of Application Server.

- **Capture Application Deployment Descriptor:** Deployment descriptors such as `ejb-jar.xml`, `sun-ejb-jar.xml`, `web.xml`, `sun-web.xml`,
- Log Level:
- Log Entries:

The number of log entries to be included in the generated diagnostic report.

Generating a Diagnostic Report

Generating a diagnostic report is based on the preferences you set in the Application Server Diagnostic tab in the Admin Console. Confidential data appears in the generated report as listed in the Confidential Properties table.

Java Virtual Machine and Advanced Settings

The Java virtual machine (JVM) is an interpretive computing engine responsible for running the byte codes in a compiled Java program. The JVM translates the Java byte codes into the native instructions of the host machine. The application server, being a Java process, requires a JVM in order to run, and to support the Java applications running on it. JVM settings are part of an application server configuration.

This chapter explains the configuration of the Java Virtual Machine (JVM™) and other advanced settings. It contains the following sections:

- “[Tuning the JVM Settings](#)” on page 243
- “[Configuring Advanced Settings](#)” on page 244

Tuning the JVM Settings

As part of configuring the application server, you define settings that enhance the use of the Java Virtual machine. To change the JVM configuration using the Admin Console, select Application Server > JVM Settings tab and define the general JVM settings as follows:

- **Java Home:** Enter the name of the installation directory of the Java software. The Application Server relies on the Java SE software. To verify that the Java SE version you specify is supported in this release, see “[Hardware and Software Requirements](#)” in *Sun Java System Application Server Platform Edition 9 Release Notes*.

Note – If you enter a nonexistent directory name or the installation directory name of an unsupported version of the Java EE software, then the Application Server will not start.

- **Javac Options:** Enter the command-line options for the Java programming language compiler. The Application Server runs the compiler when EJB components are deployed.
- **Debug:** To set up debugging with the JPDA (Java Platform Debugger Architecture), select this Enabled checkbox.

JPDA is used by application developers. For more information, see [Chapter 4, “Debugging Applications,”](#) in *Sun Java System Application Server Platform Edition 9 Developer’s Guide*.

- **Debug Options:** Specify the JPDA options passed to the JVM when the debugging is enabled.
- **RMI Compile Options:** Enter the command-line options for the `rmi c` compiler. The Application Server runs the `rmi c` compiler when EJB components are deployed.
- **Bytecode Preprocessor:** Enter a comma separated list of class names. Each class must implement the `com.sun.appserv.BytecodePreprocessor` interface. The classes are called in the order specified.

Tools such as profilers may require entries in the Bytecode Preprocessor field. Profilers generate information used to analyze server performance. For more information about profiling, see [Chapter 4, “Debugging Applications,”](#) in *Sun Java System Application Server Platform Edition 9 Developer’s Guide*.

Configuring Advanced Settings

To set the advanced application configuration using the Admin Console, select **Application Server > Advanced tab > Application Configuration tab** and set the application configuration as follows:

- **Reload:** Select this checkbox to enable dynamic reloading of applications.
If dynamic reloading is enabled (it is by default), you do not have to redeploy an application or module when you change its code or deployment descriptors. All you have to do is copy the changed JSP or 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. In a production environment, however, dynamic reloading might degrade performance. In addition, whenever a reload is done, the sessions at that transit time become invalid. The client must restart the session.
- **Reload Poll Interval:** Define the interval at which applications and modules are checked for code changes and dynamically reloaded. The default is 2.
- **Admin Session Timeout:** Specify the number of minutes of inactivity after which the admin session times out.

In addition, define the deploy settings as follows:

- **Auto Deploy:** Select this checkbox to enable automatic deployment of applications.
Autodeployment involves copying an application or module file (JAR, WAR, RAR, or EAR) into a special directory, where it is automatically deployed by the Application Server.
- **Auto Deploy Poll Interval:** Define the interval at which applications and modules are checked for code changes and dynamically reloaded. The default is 2.

- **Verifier:** Check the Verifier Enabled box to verify your deployment descriptor files. This is optional.
- **Precompile:** Check the Precompile Enabled box to precompile any JSP files.

Automatically Restarting a Domain

If your domain is stopped unexpectedly (for example, if you need to restart your machine), you can configure your system to automatically restart the domain.

This Appendix contains the following topics:

- [“Restarting Automatically on UNIX Platforms” on page 247](#)
- [“Restarting Automatically on the Microsoft Windows Platform” on page 248](#)
- [“Security for Automatic Restarts” on page 249](#)

Restarting Automatically on UNIX Platforms

To restart your domain on a UNIX platform, add a line of text to the `/etc/inittab` file.

If you use `/etc/rc.local`, or your system’s equivalent, place a line in `/etc/rc.local` that calls the desired `asadmin` command.

For example, to restart `domain1` for an Application Server installed in the `opt/SUNWappserver` directory, using a password file called `password.txt`:

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user admin  
--passwordfile /opt/SUNWappserver/password.txt domain1
```

Put the text on one line. The first three letters are a unique designator for the process and can be altered.

Restarting Automatically on the Microsoft Windows Platform

To restart automatically on Microsoft Windows, create a Windows Service. Use the `appservService.exe` and `appserverAgentService.exe` executable files shipped with the Sun Java System Application Server in conjunction with the Service Control command (`sc.exe`) provided by Microsoft.

The `sc.exe` command comes with Windows XP and is either located in the `C:\windows\system32` directory or `C:\winnt\system32` directory.

For more information on using `sc.exe`, see http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndllpro/html/msdn_scmsslite.asp.

Use `appservService.exe` and `appservAgentService.exe` as follows:

```
C:\winnt\system32\sc.exe create service-name binPath= \"fully-qualified-path-to-appservService.exe  
\"fully-qualified-path-to-asadmin.bat start-command\"  
\"fully-qualified-path-to-asadmin.bat stop-command\""  
start= auto DisplayName= \"display-name\"
```

For example, to create a service called `SunJavaSystemAppServer DOMAIN1` that starts and stops the domain `domain1`, using a password file `C:\Sun\AppServer\password.txt`:

```
C:\windows\system32\sc.exe create domain1 binPath=  
\"C:\Sun\AppServer\lib\appservService.exe \"C:\Sun\AppServer\bin\asadmin.bat  
start-domain --user admin --passwordfile C:\Sun\AppServer\password.txt domain1\"  
\"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\"" start= auto  
DisplayName= \"SunJavaSystemAppServer DOMAIN1\"
```

Note – The start and stop commands entered as part of the `binPath=` parameter must have the correct syntax. To test, run the commands from the command prompt. If the commands do not properly start or stop the domain, the service does not work correctly.

Note – Don't use a mixture of `asadmin` start and stop commands and service start and stops. Mixing the two can cause the server status to be out of sync. For example, the service might not show that the component has started even though the component is not running. To avoid this situation, always use the `sc.exe` command to start and stop the component when using services.

Security for Automatic Restarts

Handle the password and master password required when starting in one of the following ways:

- On Microsoft Windows, configure the service to ask the user for the password.
 1. In the Services Control Panel, double-click the service you created.
 2. In the Properties window, click the Log On tab.
 3. Check “Allow service to interact with desktop” to prompt for the required passwords when starting the component.

You have to log in to see the prompts, and entries are not echoed back as you type them. This method is the most secure way to use the services option, but user interaction is required before the service becomes available.

If the “interact with desktop” option is not set, the service stays in a “start-pending” state and appears to hang. Kill the service process to recover from this state.
- On Windows or UNIX, create a domain using the `--savemasterpassword=true` option and create a password file to store the admin password. When starting the component, use the `--passwordfile` option to point to the file that contains the password. The admin password can also be added by using the `--password` option of the `asadmin start` command. Be aware that this method is less secure because the admin password is stored in clear text.

For example:

1. Create domain with a saved master password. In this syntax, you are prompted for the admin password and master password:

```
asadmin create-domain --adminport 4848 --adminuser admin
--savemasterpassword=true --instanceport 8080 domain1
```

2. On Windows, create a service using a password file to populate the admin password:

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe \"C:\Sun\AppServer\bin\asadmin.bat
start-domain --user admin --passwordfile C:\Sun\AppServer\password.txt domain1\"
"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\" start= auto
DisplayName= "SJESAS_PE8.1 DOMAIN1"
```

The path to the password file `password.txt` is `C:\Sun\AppServer\password.txt`. It contains the password in the following format

```
AS_ADMIN_password=password
```

For example, for a password `adminadmin`:

```
AS_ADMIN_password=adminadmin
```

3. On UNIX, use the `--passwordfile` option in the line you add to the `inittab` file:

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user admin
--passwordfile /opt/SUNWappserver/password.txt domain1
```

The path to the password file `password.txt` is `/opt/SUNWappserver/password.txt`. It contains the password in the following format

```
AS_ADMIN_password=password
```

For example, for a password `adminadmin`:

```
AS_ADMIN_password=adminadmin
```

- Creating a service using a password that is populated from a command line option:

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe \"C:\Sun\AppServer\bin\asadmin.bat
start-domain --user admin --password adminadmin domain1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\" start= auto
DisplayName= "SJESAS_PE8.1 DOMAIN1"
```

Executing Local Admin CLI Commands without Admin Password

Sun Java System Application Server supports three local commands for which the administrator password need not be specified in clear text anymore. Instead the admin password can be defined as an alias in the passwordfile. The `asadmin` command invokes this passwordfile through "`--passwordfile`" option. This feature is also useful when a domain or a server has to be automatically restarted. The commands currently supported by this feature are `start-domain`, `start-appserv`, and `start-node-agent`.

▼ To create the password alias

- 1 Ensure that the domain is running.
- 2 Create the alias for the admin user. To do this, from the directory where the `domain.xml` file resides (`install_dir/domains/domain_dir/config` by default), run the following `asadmin` command:

```
create-password-alias --port 4848 --user admin --password adminadmin
--aliaspassword adminadmin adminalias
```

- 3 Insert the alias in a passwordfile

```
$cat passwordfile
AS_ADMIN_PASSWORD=${ALIAS=adminalias}
```

Example 20-1 for start-domain

Here's an example of the `start-domain` command used with the password alias.

```
asadmin start-domain --user admin --passwordfile passwordfile domain1
```


Dotted Name Attributes for domain.xml

This appendix describes the dotted name attributes that can be used to address the MBean and its attributes. Every element in the `domain.xml` file has a corresponding MBean. Because the syntax for using these names involves separating names between periods, these names are called *dotted names*.

This appendix contains the following topics:

- [“Top Level Elements” on page 253](#)
- [“Elements Not Aliased” on page 255](#)

Top Level Elements

The following conditions must be adhered to for all top level elements in the `domain.xml` file:

- Each server, configuration, cluster, or node agent must have a unique name.
- Servers, configurations, clusters, or node agents cannot be named `domain`.
- Server instances cannot be named `agent`.

The following table identifies the top level elements and the corresponding dotted name prefix.

Element Name	Dotted Name Prefix
applications	domain.applications
resources	domain.resources
configurations	domain.configs
servers	domain.servers
	Every server contained in this element is accessible as <i>server-name</i> . Where <i>server-name</i> is the value of the name attribute for the server subelement.

Element Name	Dotted Name Prefix
clusters	domain.clusters Every cluster contained in this element is accessible as <i>cluster-name</i> . Where <i>cluster-name</i> is the value of the name attribute for the cluster subelement.
node-agents	domain.node-agents
lb-configs	domain.lb-configs
system-property	domain.system-property

Two levels of aliasing are available:

1. The first level of alias allows access to attributes of server instances or clusters without going through the `domain.servers` or `domain.clusters` prefix. So, for example, a dotted name of the form `server1` maps to the dotted name `domain.servers.server1` (where `server1` is a server instance).
2. The second level of alias is used to refer to configurations, applications, and resources of a cluster or a standalone server instance (`target`).

The following table identifies dotted names beginning with the server name, or cluster name, that are aliased to top level names under the domain:

Dotted Name	Aliased to	Comments
<i>target.applications.*</i>	<code>domain.applications.*</code>	The alias resolves to applications referenced by the <i>target</i> only.
<i>target.resources.*</i>	<code>domain.resources.*</code>	The alias resolves to all <code>jdbc-connection-pool</code> , <code>connector-connection-pool</code> , <code>resource-adapter-config</code> , and all other resources referenced by the <i>target</i> .

The following table identifies dotted names beginning with the server name, or cluster name, that are aliased to top level names within the configuration referenced by the server or cluster.

Dotted Name	Aliased to
<i>target.http-service</i>	<i>config-name.http-service</i>
<i>target.iiop-service</i>	<i>config-name.iiop-service</i>
<i>target.admin-service</i>	<i>config-name.admin-service</i>
<i>target.web-container</i>	<i>config-name.web-container</i>
<i>target.ejb-container</i>	<i>config-name.ejb-container</i>
<i>target.mdb-container</i>	<i>config-name.mdb-container</i>

Dotted Name	Aliased to
<i>target.jms-service</i>	<i>config-name.jms-service</i>
<i>target.log-service</i>	<i>config-name.log-service</i>
<i>target.security-service</i>	<i>config-name.security-service</i>
<i>target.transaction-service</i>	<i>config-name.transaction-service</i>
<i>target.monitoring-service</i>	<i>config-name.monitoring-service</i>
<i>target.java-config</i>	<i>config-name.java-config</i>
<i>target.availability-service</i>	<i>config-name.availability-service</i>
<i>target.thread-pools</i>	<i>config-name.thread-pools</i>

Elements Not Aliased

A clustered instance should not be aliased. To get a system property for a clustered instance, the dotted name attribute you should use is as follows:

`domain.servers.clustered-instance-name.system-property`, not
`clustered-instance-name.system-property`.

The `asadmin` Utility

The Application Server includes a command-line administration utility known as `asadmin`. The `asadmin` utility is used to start and stop the Application Server, as well as manage users, resources, and applications.

This chapter contains the following sections:

- “The `asadmin` Command Usage” on page 258
- “Multimode Command” on page 261
- “The List, Get and Set Commands” on page 261
- “Server Lifecycle Commands” on page 263
- “List and Status Commands” on page 264
- “Deployment Commands” on page 264
- “Message Queue Administration Commands” on page 265
- “Resource Management Commands” on page 266
- “Application Server Configuration Commands” on page 268
- “User Management Commands” on page 271
- “Rule Commands” on page 272
- “Database Commands” on page 273
- “Diagnostic and Logging Commands” on page 273
- “Web Service Commands” on page 274
- “Security Service Commands” on page 274
- “Password Commands” on page 275
- “Verify `domain.xml` Command” on page 276
- “Custom MBean Commands” on page 276
- “Miscellaneous Commands” on page 277

The asadmin Command Usage

Use the asadmin utility to perform any administrative tasks for the Application Server. You can use this asadmin utility in place of using the Administration Console.

The asadmin utility invokes commands that identify the operation or task you wish to perform. These commands are case-sensitive. Short option arguments have a single dash (-); while long option arguments have two dashes (--). Options control how the utility performs a command. Options are also case-sensitive. Most options require argument values except boolean options which toggle to switch a feature ON or OFF. Operands appear after the argument values, and are set off by a space, a tab, or double dashes (--). The asadmin utility treats anything that comes after the options and their values as an operand.

EXAMPLE 22-1 Syntax Example

```
asadmin command [-short_option] [short_option_argument]* [--long_option  
[long_option_argument]* [operand]*
```

```
asadmin create-profiler -u admin --passwordfile password.txt myprofiler
```

To access the man pages for the Application Server asadmin utility commands on the Solaris platform, add \$AS_INSTALL/man to your MANPATH environment variable.

You can obtain overall usage information for any of the asadmin utility commands by invoking the --help option. If you specify a command, the usage information for that command is displayed. Using the --help option without a command displays a listing of all the available commands.

EXAMPLE 22-2 help Command Example

```
asadmin --help displays general help
```

```
asadmin command --help displays help for the specified command.
```

This section contains the following topics:

- [“Multi and Interactive Modes” on page 258](#)
- [“Local Commands” on page 259](#)
- [“Remote Commands” on page 259](#)
- [“The Password File” on page 261](#)

Multi and Interactive Modes

The asadmin utility can be used in command shell invocation or multi command mode (known as the multimode command). In command shell invocation you invoke the asadmin utility from your command shell. asadmin executes the command, then exits. In multiple command mode,

you invoke `asadmin` once, it then accepts multiple commands until you exit `asadmin` and return to the normal command shell invocation. Environment variables set while in multiple command mode are used for all subsequent commands until you exit `multimode`. You may provide commands by passing a previously prepared list of commands from a file or standard input (pipe). Additionally, you can invoke `multimode` from within a multimode session; once you exit the second multimode environment, you return to your original multimode environment.

You can also run the `asadmin` utility in interactive or non-interactive mode. By default, the interactive mode option is enabled. It prompts you for the required arguments. You can use the interactive mode option in command shell invocation under all circumstances. You can use the interactive mode option in `multimode` when you run one command at a time from the command prompt; and when you run in `multimode` from a file. Commands in `multimode`, when piped from an input stream, and commands invoked from another program, cannot run in the interactive mode.

Local Commands

Local commands can be executed without the presence of an administration server. However, it is required that the user be logged into the machine hosting the domain in order to execute the command and have access (permissions) for the installation and domain directories.

For commands that can be executed locally or remotely, if any one of the `--host`, `--port`, `--user`, and `--passwordfile` options are set, either in the environment or in the command line, the command will run in remote mode. Also, if none of the local options are set, either on the command line or in the environment, the command is executed locally by default. For commands that can be executed locally or remotely, if any one of the `--host`, `--port`, `--user`, and `--passwordfile` options are set, either in the environment or in the command line, the command will run in remote mode.

Remote Commands

Remote commands are always executed by connecting to an administration server and executing the command there. A running administration server is required. All the remote commands require the following common options:

TABLE 22-1 Remote Commands Required Options

Short Option	Option	Definition
-H	--host	The machine name where the domain administration server is running. The default value is localhost.

TABLE 22-1 Remote Commands Required Options (Continued)

Short Option	Option	Definition
-p	--port	The HTTP/S port for administration. This is the port to which you should point your browser in order to manage the domain. For example, <code>http://localhost:4848</code> . The default port number for Platform Edition is 4848.
-u	--user	The authorized domain administration server administrative username. If you have authenticated to a domain using the <code>asadmin login</code> command, then you need not specify the <code>--user</code> option on subsequent operations to this particular domain.
	--passwordfile	<p>The <code>--passwordfile</code> option specifies the name of a file containing the password entries in a specific format. The entry for the password must have the <code>AS_ADMIN_</code> prefix followed by the password name in uppercase letters.</p> <p>For example, to specify the domain administration server password, use an entry with the following format: <code>AS_ADMIN_PASSWORD=password</code>, where <i>password</i> is the actual administrator password. Other passwords that can be specified include <code>AS_ADMIN_PASSWORD</code>, <code>AS_ADMIN_USERPASSWORD</code>, and <code>AS_ADMIN_ALIASEXPASSWORD</code>, <code>AS_ADMIN_MAPPEDPASSWORD</code>.</p> <p>All remote commands must specify the admin password to authenticate to the domain administration server, either through <code>--passwordfile</code> or <code>asadmin login</code>, or interactively on the command prompt. The <code>asadmin login</code> command can be used only to specify the admin password. For other passwords, that must be specified for remote commands, use the <code>--passwordfile</code> or enter them at the command prompt.</p> <p>If you have authenticated to a domain using the <code>asadmin login</code> command, then you need not specify the admin password through the <code>--passwordfile</code> option on subsequent operations to this particular domain. However, this is applicable only to <code>AS_ADMIN_PASSWORD</code> option. You will still need to provide the other passwords, for example, <code>AS_ADMIN_USERPASSWORD</code>, as and when required by individual commands, such as <code>update-file-user</code>.</p> <p>For security reasons, passwords specified as an environment variable will not be read by <code>asadmin</code>.</p>
-s	--secure	If set to true, uses SSL/TLS to communicate with the domain administration server.
-I	--interactive	If set to true (default), only the required password and user options are prompted.
-t	--terse	Indicates that any output data must be very concise, typically avoiding human-friendly sentences and favoring well-formatted data for consumption by a script. Default is false.

TABLE 22-1 Remote Commands Required Options (Continued)

Short Option	Option	Definition
-e	--echo	Setting to true will echo the command line statement on the standard output. Default is false.
-h	--help	Displays the help text for the command.

The Password File

For security purposes, you can set the password for a command from a file instead of entering the password at the command line. The `--passwordfile` option takes the file containing the passwords. The valid contents for the file are:

EXAMPLE 22-3 Passwordfile contents

```
AS_ADMIN_PASSWORD=value
AS_ADMIN_ADMINPASSWORD=value
AS_ADMIN_USERPASSWORD=value
AS_ADMIN_MASTERPASSWORD=value
```

Multimode Command

Use the `multimode` command to process the `asadmin` commands. The command-line interface will prompt you for a command, execute that command, display the results of the command, and then prompt you for the next command. Additionally, all the `asadmin` option names set in this mode are used for all the subsequent commands. You can set your environment and run commands until you exit `multimode` by typing “exit” or “quit.” You can also provide commands by passing a previously prepared list of commands from a file or standard input (pipe). You can invoke `multimode` from within a *multimode* session; once you exit the second *multimode* environment, you return to your original *multimode* environment.

To invoke `multimode`, enter `asadmin multimode`.

The List, Get and Set Commands

The `asadmin list`, `get` and `set` commands work in tandem to provide a navigation mechanism for the Application Server's dotted naming hierarchy. There are two hierarchies: **configuration** and **monitoring** and these commands operate on both. The `list` command provides the fully qualified dotted names of the management components that have read-only or modifiable attributes.

The **configuration** hierarchy provides attributes that are modifiable; whereas the attributes of management components from the **monitoring** hierarchy are purely read-only.

The **configuration** hierarchy is loosely based on the domain's schema document; whereas the **monitoring** hierarchy is a little different.

Use the `list` command to reach a particular management component in the desired hierarchy. Then, invoke the `get` and `set` commands to get the names and values or set the values of the attributes of the management component at hand. Use the wildcard (*) option to fetch all matches in a given fully qualified dotted name.

An Application Server dotted name uses the "." (period) as a delimiter to separate the parts of a complete name. This is similar to how the "/" character is used to delimit the levels in the absolute path name of a file in the UNIX file system. The following rules apply while forming the dotted names accepted by the `get`, `set`, and `list` commands. Note that a specific command has some additional semantics applied.

- A . (period) always separates two sequential parts of the name.
- A part of the name usually identifies an application server subsystem and/or its specific instance. For example: `web-container`, `log-service`, `thread-pool-1`, etc.
- If any part of the name itself contains a . (period), then it must be escaped with a leading \ (backslash) so that the "." does not act like a delimiter.
- An * (asterisk) can be used anywhere in the dotted name and it acts like the wildcard character in regular expressions. Additionally, an * can collapse all the parts of the dotted name. Long dotted name like "`<classname>this.is.really.long.hierarchy</classname>`" can be abbreviated to "`<classname>th*.hierarchy</classname>`." But note that the . always delimits the parts of the name.
- On Solaris, quotes are needed when executing commands with * as the option value or operand.
- The top level switch for any dotted name is `--monitor` or `-m` that is separately specified on a given command line. The presence or lack of this switch implies the selection of one of the two hierarchies for application server management: monitoring and configuration.
- If you happen to know the exact complete dotted name without any wildcard character, then `list` and `get/set` have a little difference in their semantics:
 - The `list` command treats this complete dotted name as the complete name of a parent node in the hierarchy. Upon providing this name to the `list` command, it simply returns the names of the immediate children at that level. For example, `list server.applications.web-module` will list all the web modules deployed to the domain or the default server.
 - The `get` and `set` commands treat this complete dotted name as the fully qualified name of the attribute of a node (whose dotted name itself is the name that you get when you remove the last part of this dotted name) and it gets/sets the value of that attribute. This is true if such an attribute exists. You will never start with this case because in order to find out the names of attributes of a particular node in the hierarchy, you must use the

wildcard character *. For example, `server.applications.web-module.JSPWiki.context-root*` returns the context-root of the web-application deployed to the domain or default server.

The `list` command is the progenitor of navigational capabilities of these three commands. If you want to set or get attributes of a particular application server subsystem, you must know its dotted name. The `list` command is the one which can guide you to find the dotted name of that subsystem. For example, to find out the modified date (attribute) of a particular file in a large file system that starts with /. First you must find out the location of that file in the file system, and then look at its attributes. Therefore, two of the first commands to understand the hierarchies in the Application Server are: `* list "*"` and `<command>* list * --monitor`. Consult the `get`, `set` or `list` commands man pages to identify the sorted output of these commands.

Server Lifecycle Commands

The server lifecycle commands are commands that create, delete, or start, stop a domain, service (DAS), or an instance.

TABLE 22-2 Server Lifecycle Commands

Command	Definition
<code>create-service</code>	Configures the starting of a DAS on an unattended boot. On Solaris 10, this command uses the Service Management Facility (SMF). This is a local command and must be run as the OS-level user with superuser privileges. It is available only for Solaris 10. When the service is created, the user has to start, enable, disable, delete, or stop the service. The DAS must be stored on a folder to which the super-user has access. The configuration cannot be stored on a network file system. The service is created such that it is controlled by the OS-level user, who owns the folder where the configuration of the DAS resides. To run this command, you must have <code>solaris.smf.*</code> authorization.
<code>create-domain</code>	Creates the configuration of a domain. A domain is an administrative namespace. Every domain has a configuration, which is stored in a set of files. Any number of domains, each of which has a distinct administrative identity, can be created in a given installation of the Application Server. A domain can exist independent of other domains. Any user who has access to the <code>asadmin</code> utility on a given system can create a domain and store its configuration in a folder of choice. By default, the domain configuration is created in the <code>install_dir/domains</code> directory. You can override this location to store the configuration elsewhere.
<code>delete-domain</code>	Deletes the named domain. The domain must already exist and must be stopped.

TABLE 22-2 Server Lifecycle Commands (Continued)

Command	Definition
start-domain	Starts a domain. If the domain directory is not specified, the domain in the default <i>install_dir/domains</i> directory is started. If there are two or more domains, the <i>domain_name</i> operand must be specified.
stop-domain	Stops the Domain Administration Server of the specified domain.
restore-domain	Restores files under the domain from a backup directory.
list-domains	Lists the domain. If the domain directory is not specified, the domain in the default <i>install_dir/domains</i> directory is listed. If there is more than one domain, the <i>domain_name</i> operand must be specified.
backup-domain	Backs up files under the named domain.
list-backups	Displays the status information about all backups in the backup repository.
shutdown	Gracefully brings down the administration server and all the running instances. You must manually start the administration server to bring it up again.

List and Status Commands

The list and status commands display the status of a deployed component.

TABLE 22-3 List and Status Commands

Command	Definition
show-component-status	Gets the status of the deployed component. The status is a string representation returned by the server. The possible status strings include status of <i>app-name</i> is enabled or status of <i>app-name</i> is disabled.
list-components	Lists all deployed Java EE 5 components. If the <i>-type</i> option is not specified, all components are listed.
list-sub-components	Lists EJBs or Servlets in a deployed module or in a module of the deployed application. If a module is not identified, all modules are listed.

Deployment Commands

The deployment commands deploy an application or get the client stubs.

TABLE 22-4 Deployment Commands

Command	Definition
<code>deploy</code>	Deploys an enterprise application, web application, EJB module, connector module, or application client module. If the component is already deployed or already exists, it is forcefully redeployed if the <code>--force</code> option is set to <code>true</code> .
<code>deploydir</code>	Deploys an application directly from a development directory. The appropriate directory hierarchy and deployment descriptors conforming to the Java EE specification must exist in the deployment directory.
<code>get-client-stubs</code>	Gets the client stubs JAR file for an <code>AppClient</code> standalone module or an application containing the <code>AppClient</code> module, from the server machine to the local directory. The application or module should be deployed before executing this command. It is also possible to get the client stubs as part of the <code>deploy</code> command using the <code>--retrieve</code> option.
<code>undeploy</code>	Removes the specified deployed component.

Message Queue Administration Commands

The Message Queue administration commands allow you to manage the JMS destinations.

TABLE 22-5 Message Queue Commands

Command	Definition
<code>create-jmsdest</code>	Creates a JMS physical destination. Along with the physical destination, you use the <code>create-jms-resource</code> command to create a JMS destination resource that has a <code>Name</code> property that specifies the physical destination.
<code>delete-jmsdest</code>	Removes the specified JMS destination.
<code>flush-jmsdest</code>	Purges the messages from a physical destination in the specified target's JMS Service configuration.
<code>list-jmsdest</code>	Lists the JMS physical destinations.
<code>jms-ping</code>	Checks if the JMS service (also known as the JMS provider) is up and running. When you start the Application Server, the JMS service starts by default. Additionally, it pings only the default JMS host within the JMS service. It displays an error message when it is unable to ping a built-in JMS service.

Resource Management Commands

The resource commands allow you to manage the various resources used in your application.

TABLE 22-6 Resource Management Commands

Command	Definition
create-jdbc-connection-pool	Registers a new JDBC connection pool with the specified JDBC connection pool name.
delete-jdbc-connection-pool	Deletes a JDBC connection pool. The operand identifies the JDBC connection pool to be deleted.
list-jdbc-connection-pools	Gets the JDBC connection pools that have been created.
create-jdbc-resource	Creates a new JDBC resource.
delete-jdbc-resource	Removes a JDBC resource with the specified JNDI name.
list-jdbc-resources	Displays a list of JDBC resources that have been created.
create-jms-resource	Creates a Java Message Service (JMS) connection factory resource or a JMS destination resource.
delete-jms-resource	Removes the specified JMS resource.
list-jms-resources	Lists the existing JMS resources (destination and connection factory resources).
create-jndi-resource	Registers a JNDI resource.
delete-jndi-resource	Removes the JNDI resource with the specified JNDI name.
list-jndi-resources	Identifies all the existing JNDI resources.
list-jndi-entries	Browses and queries the JNDI tree.
create-javamail-resource	Creates a JavaMail session resource.
delete-javamail-resource	Removes the specified JavaMail session resource.
list-javamail-resources	Lists the existing JavaMail session resources.
create-persistence-resource	Registers a persistence resource.
delete-persistence-resource	Removes a persistence resource. When you delete a persistence resource, the command also removes the JDBC resource if it was created using the create-persistence-resource command.
list-persistence-resources	Displays all the persistence resources.
create-custom-resource	Creates a custom resource. A custom resource specifies a custom server-wide resource object factory that implements the javax.naming.spi.ObjectFactory interface.

TABLE 22-6 Resource Management Commands (Continued)

Command	Definition
<code>delete-custom-resource</code>	Removes a custom resource.
<code>list-custom-resources</code>	Lists the custom resources.
<code>create-connector-connection-pool</code>	Adds a new connector connection pool with the specified connection pool name.
<code>delete-connector-connection-pool</code>	Removes the connector connection pool specified using the operand <code>connector_connection_pool_name</code> .
<code>list-connector-connection-pools</code>	Lists the connector connection pools that have been created.
<code>create-connector-resource</code>	Registers the connector resource with the specified JNDI name.
<code>delete-connector-resource</code>	Removes the connector resource with the specified JNDI name.
<code>list-connector-resources</code>	Gets all the connector resources.
<code>create-admin-object</code>	Adds the administered object that has the specified JNDI name.
<code>delete-admin-object</code>	Removes the administered object with the specified JNDI name.
<code>list-admin-objects</code>	Lists all the administered objects.
<code>create-resource-adapter-config</code>	Creates configuration information for the connector module.
<code>delete-resource-adapter-config</code>	Deletes the configuration information created in <code>domain.xml</code> for the connector module.
<code>list-resource-adapter-configs</code>	Lists the configuration information in the <code>domain.xml</code> for the connector module
<code>add-resources</code>	Creates the resources named in the specified XML file. The <i>xml_file_path</i> is the path to the XML file containing the resources to be created. The DOCTYPE should be specified as <i>install_dir/lib/dtds/sun-resources_1_2.dtd</i> in the <code>resources.xml</code> file.
<code>ping-connection-pool</code>	Tests if a connection pool is usable for both JDBC connection pools and connector connection pools. For example, if you create a new JDBC connection pool for an application that is expected to be deployed later, the JDBC pool is tested with this command before deploying the application. Before pinging a connection pool, you must create the connection pool with authentication and ensure that the Application Server or database is started.

Application Server Configuration Commands

The configuration commands allow you to configure the operation of the Application Server. This section contains the following topics:

- [“General Configuration Commands” on page 268](#)
- [“HTTP, IIOP and SSL Listener Commands” on page 268](#)
- [“Lifecycle and Audit Module Commands” on page 269](#)
- [“Profiler and JVM Options Commands” on page 270](#)
- [“Virtual Server Commands” on page 270](#)
- [“Threadpool Commands” on page 270](#)
- [“Transaction and Timer Commands” on page 271](#)

General Configuration Commands

These commands allow you to manage the configuration of the Application Server components.

TABLE 22-7 General Configuration Commands

Command	Definition
enable	Enables the specified component. If the component is already enabled, then it is re-enabled. The component must have been deployed in order to be enabled. If it has not been deployed, then an error message is returned.
disable	Immediately disables the named component. The component must have been deployed. If the component has not been deployed, an error message is returned.
export	Marks a variable name for automatic export to the environment of subsequent commands. All subsequent commands use the variable name value as specified unless you unset them or exit multimode.
get	Gets the names and values of attributes.
set	Sets the values of one or more configurable attribute.
list	Lists the configurable element. On Solaris, quotes are needed when executing commands with * as the option value or operand.
unset	Removes one or more variables you set for the multimode environment. The variables and their associated values will no longer exist in the environment.

HTTP, IIOP and SSL Listener Commands

The HTTP and IIOP listener commands help you manage listeners. These commands are supported in remote mode only.

TABLE 22-8 IIOP Listener Commands

Command	Definition
<code>create-http-listener</code>	Adds a new HTTP listener.
<code>delete-http-listener</code>	Removes the specified HTTP listener.
<code>list-http-listeners</code>	Lists the existing HTTP listener.
<code>create-iiop-listener</code>	Creates an IIOP listener.
<code>delete-iiop-listener</code>	Removes the specified IIOP listener.
<code>list-iiop-listeners</code>	Lists the existing IIOP listeners.
<code>create-ssl</code>	Creates and configures the SSL element in the selected HTTP listener, IIOP listener, or IIOP service to enable secure communication on that listener/service.
<code>delete-ssl</code>	Deletes the SSL element in the selected HTTP listener, IIOP listener, or IIOP service.

Lifecycle and Audit Module Commands

The lifecycle and audit module commands help you control lifecycle modules and optional plug-in modules which implement audit capabilities. The commands are supported in remote mode only.

TABLE 22-9 Lifecycle Module Commands

Command	Definition
<code>create-lifecycle-module</code>	Creates a lifecycle module. The lifecycle modules provide a means of running short or long duration Java-based tasks within the Application Server environment.
<code>delete-lifecycle-module</code>	Removes the specified lifecycle module.
<code>list-lifecycle-modules</code>	Lists the existing lifecycle module.
<code>create-audit-module</code>	Adds the named audit module for the plug-in module that implements the audit capabilities.
<code>delete-audit-module</code>	Removes the named audit module.
<code>list-audit-modules</code>	Lists all the audit modules.

Profiler and JVM Options Commands

The Profiler and JVM options commands allow you to administrate profilers and control these elements. These commands are supported in remote mode only.

TABLE 22-10 Profiler and JVM Options Commands

Command	Definition
create-profiler	Creates the profiler element. A server instance is tied to a particular profiler, by the profiler element in the Java configuration. Changing a profiler requires you to restart the server.
delete-profiler	Deletes the profiler element you specify. A server instance is tied to a particular profiler by the profiler element in the Java configuration. Changing a profiler requires you to restart the server.
create-jvm-option	Creates JVM options in the Java configuration or profiler elements of the <code>domain.xml</code> file. If JVM options are created for a profiler, they are used to record the settings needed to get a particular profiler going. You must restart the server for newly created JVM options to take effect.
delete-jvm-option	Removes JVM options from the Java configuration or profiler elements of the <code>domain.xml</code> file.

Virtual Server Commands

The Virtual Server commands allow you to control these elements. These commands are supported in remote mode only.

TABLE 22-11 Virtual Server Commands

Command	Definition
create-virtual-server	Creates the named virtual server. Virtualization in the Application Server allows multiple URL domains to be served by a single HTTP server process that is listening on multiple host addresses. If the application is available at two virtual servers, they still share the same physical resource pools.
delete-virtual-server	Removes the virtual server with the specified virtual server ID.
list-virtual-server	Lists the existing virtual servers.

Threadpool Commands

The threadpool commands allow you to control these elements. These commands are supported in remote mode only.

TABLE 22-12 Threadpool Commands

Command	Definition
<code>create-threadpool</code>	Creates a threadpool with the specified name. You can specify maximum and minimum number of threads in the pool, the number of work queues, and the idle timeout of a thread. The created thread pool can be used for servicing IIOP requests and for resource adapters to service work management requests. A created thread pool can be used in multiple resource adapters.
<code>delete-threadpool</code>	Removes the threadpool with the named ID.
<code>list-threadpools</code>	Lists all the thread pools.

Transaction and Timer Commands

The transaction and timer commands allow you to control the transaction and timer subsystems; allowing you to suspend any inflight transactions. These commands are supported in remote mode only.

TABLE 22-13 Transaction Commands

Command	Definition
<code>freeze-transaction</code>	Freezes the transaction subsystem during which time all the inflight transactions are suspended. Invoke this command before rolling back any inflight transactions. Invoking this command on an already frozen transaction subsystem has no effect.
<code>unfreeze-transaction</code>	Resumes all the suspended inflight transactions. Invoke this command on an already frozen transaction.
<code>recover-transactions</code>	Manually recovers pending transactions.
<code>rollback-transaction</code>	Rolls back the named transaction.
<code>list-timers</code>	Lists the timers owned by a specific server instance

User Management Commands

These user commands are to administer the users support by the file realm authentication. These commands are supported in remote mode only.

TABLE 22-14 User Management Commands

Command	Definition
create-file-user	Creates an entry in the keyfile with the specified username, password, and groups. Multiple groups can be created by separating them with a colon (:).
delete-file-user	Deletes the entry in the keyfile with the specified username.
update-file-user	Updates an existing entry in the keyfile using the specified user_name, user_password and groups. Multiple groups can be entered by separating them, with a colon (:).
list-file-users	Creates a list of file users supported by file realm authentication.
list-file-groups	Lists the users and groups supported by the file realm authentication. This command lists available groups in the file user.

Monitoring Data Commands

The monitoring data commands allow you to monitor the server. These commands are supported in remote mode only.

TABLE 22-15 Monitoring Data Commands

Command	Definition
start-callflow-monitoring	Collects and correlates data from Web container, EJB container and JDBC to provide a complete call flow/path of a request. Data is collected only if callflow-monitoring is ON.
stop-callflow-monitoring	Disables collection of call flow information of a request.

Rule Commands

Rules commands allow you to manage rules for the server. These commands are supported in remote mode only.

TABLE 22-16 Rules Commands

Command	Definition
create-management-rule	Creates a new management rule to intelligently self-manage the Application Server installation and deployed applications.
delete-management-rule	Removes the management rule you specify.

TABLE 22-16 Rules Commands *(Continued)*

Command	Definition
list-management-rules	Lists the available management rules.

Database Commands

The database commands allow you to start and stop the Java DB database (based on Apache Derby). These commands are supported in local mode only.

TABLE 22-17 Database Commands

Command	Definition
start-database	Starts the Java DB server that is available with the Application Server. Use this command only for working with applications deployed to the Application Server.
stop-database	Stops a process of the Java DB server. Java DB server is available with the Application Server.

Diagnostic and Logging Commands

The diagnostic and logging commands help you troubleshoot problems with the Application Server. These commands are supported in remote mode only.

TABLE 22-18 Diagnostic and Logging Commands

Command	Definition
generate-diagnostic-report	Generates an HTML report that contains pointers or navigational links to an application server installation details such as configuration details, logging details, or process specific information for an application server instance.
generate-jvm-report	Displays the threads (dump of stack trace), classes and memory for a given target instance, including the Domain Administration Service. This command works only with the application server instance processes. This command replaces the traditional techniques like sending <code>ctrl+break</code> or <code>kill -3</code> signals to application server processes. The command will not work if the target server instance is not running.
display-error-statistics	Displays a summary list of severities and warnings in <code>server.log</code> since the last server restart.
display-error-distribution	Displays distribution of errors from instance <code>server.log</code> at module level. TM

TABLE 22-18 Diagnostic and Logging Commands (Continued)

Command	Definition
display-log-records	Displays all the error messages for a given module at a given timestamp.

Web Service Commands

The web service commands allow you to monitor a deployed web service and manage transformation rules.

TABLE 22-19 Web Service Commands

Command	Definition
configure-webservice-management	Configure the monitoring or the maxhistorysize attributes of a deployed web service endpoint.
create-transformation-rule	Creates an XSLT transformation rule that can be applied to a web service operation. The rule can be applied to a request, a response, or both.
delete-transformation-rule	Deletes an XSLT transformation rule of a given web service.
list-transformation-rules	Lists all the transformation rules of a given web service in the order they are applied.
publish-to-registry	Publishes the web service artifacts to registry servers.
unpublish-from-registry	Unpublishes the web service artifacts from the registry servers.
list-registry-locations	Displays a list of configured web service entry access points.

Security Service Commands

These security commands are used to control the security mapping for the connector connection pool. These commands are supported in remote mode only.

TABLE 22-20 Security Commands

Command	Definition
<code>create-connector-security-map</code>	Creates a security map for the specified connector connection pool. If the security map is not present, a new one is created. Also, use this command to map the caller identity of the application (principal or user group) to a suitable enterprise information system (EIS) principal in container-managed transaction-based scenarios. One or more named security maps may be associated with a connector connection pool. The connector security map configuration supports the use of the wild card asterisk (*) to indicate all users or all user groups. For this command to succeed, you must have first created a connector connection pool. The EIS is any system that holds the data of an organization. It can be a mainframe, a messaging system, a database system, or an application.
<code>delete-connector-security-map</code>	Deletes a security map for the specified connector connection pool.
<code>update-connector-security-map</code>	Modifies a security map for the specified connector connection pool.
<code>list-connector-security-maps</code>	Lists the security maps belonging to the specified connector connection pool.
<code>create-message-security-provider</code>	Enables administrators to create a <code>provider-config</code> sub-element for the given message layer (<code>message-security-config</code> element of <code>domain.xml</code> , the file that specifies parameters and properties to the Application Server).
<code>delete-message-security-provider</code>	Enables administrators to delete a <code>provider-config</code> sub-element for the given message layer (<code>message-security-config</code> element of <code>domain.xml</code> , the file that specifies parameters and properties to the Application Server).
<code>list-message-security-providers</code>	Enables administrators to list all security message providers (<code>provider-config</code> sub-elements) for the given message layer (<code>message-security-config</code> element of <code>domain.xml</code>).
<code>create-auth-realm</code>	Adds the named authentication realm.
<code>delete-auth-realm</code>	Removes the named authentication realm.
<code>list-auth-realms</code>	Lists the existing authentication realms.

Password Commands

The password commands allow you to manage passwords and ensure security for the Application Server.

TABLE 22-21 Password Commands

Command	Definition
create-password-alias	Creates an alias for a password and stores it in domain.xml. An alias is a token of the form \${ALIAS=password-alias-password}. The password corresponding to the alias name is stored in an encrypted form. This command takes both a secure interactive form (in which the user is prompted for all information) and a more script-friendly form, in which the password is propagated on the command line.
delete-password-alias	Deletes a password alias.
update-password-alias	Updates the password alias IDs in the named target.
list-password-aliases	Lists all password aliases.
change-admin-password	This remote command modifies the admin password. This command is interactive in that the user is prompted for the old and new admin password (with confirmation).
change-master-password	This local command is used to modify the master password. This command is interactive in that the user is prompted for the old and new master password. This command will not work unless the server is stopped.

Verify domain.xml Command

The XML verify command verifies the content of the domain.xml file.

TABLE 22-22 Verify domain.xml Command

Command	Definition
verify-domain-xml	Verifies the content of the domain.xml file.

Custom MBean Commands

The MBean commands allow you to manage and register custom MBeans. The commands are supported in remote mode only.

TABLE 22-23 Custom MBean Commands

Command	Definition
create-mbean	Creates and registers a custom MBean . If the target MBeanServer (DAS) is not running, the MBean is not registered.

TABLE 22–23 Custom MBean Commands (Continued)

Command	Definition
<code>delete-mbean</code>	Deletes a custom MBean. Ensure that the target MBeanServer is running.
<code>list-mbeans</code>	Lists the custom MBeans for the specified target.

Miscellaneous Commands

The miscellaneous commands allow you to manage various aspects of the Application Server.

TABLE 22–24 Miscellaneous Commands

Command	Definition
<code>login</code>	Lets you log into a domain. If various application server domains are created on various machines (locally), <code>asadmin</code> invocation from any of these machines can manage the domains located elsewhere (remotely). This comes in handy especially when a particular machine is chosen as an administration client and it manages multiple domains and servers. <code>asadmin</code> commands that are used to manage domains located elsewhere are called remote commands. The <code>asadmin login</code> command eases the administration of such remote domains. The <code>login</code> command runs only in the interactive mode. It prompts you for the admin user name and password. On successful login, the file <code>.asadminpass</code> will be created in the user's home directory. This is the same file that is modified by the <code>create-domain</code> command while using the <code>--saveLogin</code> option. The domain must be running for this command to run.
<code>version</code>	Displays the version information. If the command cannot communicate with the administration server with the given user/password and host/port, then the command will retrieve the version locally and display a warning message.
<code>help</code>	Displays a list of all the <code>asadmin</code> utility commands. Specify the command to display the usage information for that command
<code>install-license</code>	Prevents unauthorized use of the Application Server. Allows you to install the license file.

Index

A

ACC

- See containers

- application client, 81

- acceptor threads, in HTTP listeners, 163

- Admin Console, 34

- applets, 81

- Application Server

- restart, 87

- shut down, 87

- Application Server domains, 36

- applications, performance, 85

- asadmin command, 180, 181

- create-threadpool, 180

- delete-threadpool, 181

- asadmin utility, 34

B

- bean-cache, monitoring attribute names, 199-200

C

- cache-hits, 199

- cache-misses, 199

- caching

- cleanup, 86

- disabling, 85

- Enterprise JavaBeans, 85

- timeouts, 86

- client access, 32

- CloudScape JDBC driver, 44-45

- Connection, Factories, JMS, 57

- connection factories, JMS, overview, 55-57

- connector, 33

- connector connection pools, JMS resources and, 57

- connector resources, JMS resources and, 57

- connectors, modules, 179

- container, 32

- containers

- applet, 81

- application client, 81

- Enterprise JavaBeans, 81, 82, 85-87

- configuring, 85-87

- Java EE, 81

- servlet

- See containers, 81

- web, 81

- web, 81

- CORBA, 177

- threads, 179

- create-domain command, 36

- custom resources, using, 75

D

- databases

- DB2, 88

- Java DB

- Derby, 88

- JNDI names, 73

databases (*Continued*)

- MS SQL Server, 88
 - Oracle, 88
 - PointBase, 88
 - resource references, 74
 - supported, 43
 - Sybase, 88
- delete-domain command, 37
- Derby JDBC driver, 44-45
- Destination
- Physical, JMS, 58
 - Resources, JMS, 58
- destinations, JMS, overview, 55-57
- documentation, overview, 23-24
- domains, creating, 36

E

- ebXML, 166
- Enterprise Java Beans, threads, 179
- Enterprise JavaBeans
- activating, 82
 - active, 86
 - authorization, 82
 - caching, 82, 85, 86-87
 - creating, 82
 - entity, 82, 85, 86-87
 - idle, 85, 86
 - message-driven, 82, 87
 - passivating, 82, 85, 86
 - persistent, 82
 - pooling, 85-86, 87
 - removing from cache, 86
 - removing idle, 87
 - session, 82
 - stateful session, 86-87, 87
 - stateless session, 85
 - timer service, 87-88
- entity beans
- See Enterprise JavaBeans
 - entity, 85
- execution-time-millis, 197
- external repositories, accessing, 75

F

- Foreign Providers, JMS, 60-69

G

- get command, monitoring data, 218

H

- HTTP listeners
- acceptor threads, 163
 - default virtual server, 163
 - overview, 162-164
- HTTP service
- HTTP listeners, 162-164
 - virtual servers, 161-162
- HTTP sessions, 83

I

- IBM DB2 JDBC driver, 45, 47
- IIOP listeners, 178
- Inet MSSQL JDBC driver, 52
- Inet Oracle JDBC driver, 51-52
- Inet Sybase JDBC driver, 53
- Informix Type 4 JDBC driver, 54

J

- Java API for XML-based remote procedure calls (JAX-RPC), 167
- Java API for XML Web Services (JAX-WS) 2.0, 167
- Java DB JDBC driver, 44-45
- Java Message Service (JMS), See JMS resources, 55
- Java Naming and Directory Service, See JNDI, 82
- Javadoc, 24
- JavaEE group, 115
- JavaMail, 33
- JavaServer Pages, 81
- JAX-RPC, 167
- JAX-WS 2.0, 167

JB1 providers, Web services as, 176

JCE provider
 configuring, 142

JDBC, 33
 drivers, 156
 resources, 88
 supported drivers, 43

JMS
 Connection Factories, 57
 Destination Resources, 58
 Foreign Providers, 60-69
 Physical Destination, 58
 providers, 59-60
 Resource Adapter, Generic, 60-69

jms-max-messages-load, 199

JMS provider, 55

JMS resources
 connection factory resources, 55-57
 destination resources, 55-57
 overview, 55-57
 physical destinations, 55-57
 queues, 55-57
 topics, 55-57

jmsra system resource adapter, 57

JNDI, 82
 custom resources, using, 75
 external repositories, 75
 lookups and associated references, 74
 names, 73, 88

JSP, See JavaServer Pages, 81

JSR 109, 167

JSR 181, 167

K

keystore.jks file, 127-128
keypoint intervals, 159
keypoint operations, 159

L

list command, monitoring, 217
list-domains command, 37

log levels, configuring, 186

log records, 183-184

logging
 configuring general settings, 186
 configuring levels, 186
 logger namespaces, 184-186
 overview, 183-184
 viewing the server log, 187-189

M

man pages, 34

Message Queue software, 55

Messaging, 33

MM MySQL Type 4 JDBC driver
 non-XA, 49-50
 XA only, 50-51

monitoring
 bean-cache attributes, 199-200
 container subsystems, 193
 ORB service, 206
 transaction service, 207
 using get command, 218
 using list command, 217
MSSQL Inet JDBC driver, 52
MSSQL/SQL Server2005 Data Direct JDBC driver, 46
MSSQL/SQL Server2005 JDBC driver, 48-49

N

naming, JNDI and resource reference, 74
naming and directory service, 33
naming service, 33
numbeansinpool, 199
numexpiredsessionsremoved, 200
numpassivationerrors, 200
numpassivations, 200
numpassivationsuccess, 200
numthreadswaiting, 199

O

Oasis Web Services Security, *See* WSS
object request broker, threads, 179
Object Request Broker (ORB), 177
 overview, 178
online help, 36
Open ESB, 176
Oracle, 88
Oracle Data Direct JDBC driver, 45-46
Oracle Inet JDBC driver, 51-52
Oracle OCI JDBC driver, 53-54
Oracle Thin Type 4 Driver, workaround for, 157
Oracle Thin Type 4 JDBC driver, 47-48
oracle-xa-recovery-workaround property, 157
ORB, 177
 IIOP listeners, 178
 overview, 178
 See object request broker, 179
 service, monitoring, 206

P

performance
 increasing, 85
 problems, 85
 thread pools, 179
Physical Destination, JMS, 58
PointBase, 88
pooling
 Enterprise JavaBeans, 85-86, 87
Port listeners, 38
PostgreSQL JDBC driver, 49
Providers, JMS, 59-60

Q

queues
 work
 See thread pools, 180
queues, JMS, 55-57

R

realms, certificate, 100
reap interval, 83
Registry, web service, 170
Registry, Web service
 adding, 173
 adding connector module for, 170
 configuring resource adapter for, 171
Registry, web service, creating, 171
Resource Adapter, Generic, JMS, 60-69
resource adapters, 156
 jmsra, 57
resource managers, 156
resource references, 74
rollback
 See transactions
 rolling back, 156
RSA encryption, 142

S

SAAJ, 167
security, 33
server administration, 33
server log, viewing, 187-189
services, timer, 87-88
services for applications, 33
servlets, 81
session manager, 83
sessions
 configuring, 83-84
 custom IDs, 83
 deleting, 84
 deleting data, 83
 file name, 83
 HTTP, 83, 85
 IDs, 83
 inactive, 83, 84
 managing, 83
 storing, 85
 storing data, 83
Simple Object Access Protocol (SOAP), 166
SOAP, 166
SOAP with Attachments API for Java (SAAJ), 167

start-domain command, 37
 stateful session beans, See Enterprise JavaBeans, 86-87
 stateless session beans, See Enterprise JavaBeans, 85
 stop-domain command, 38
 Sun Java System Message Queue software, 55
 Sybase Data Direct JDBC driver, 46-47
 Sybase Inet JDBC driver, 53
 Sybase JConnect Type 4 JDBC driver, 51

T

thread pools, 179
 creating, 180
 deleting, 181
 editing, 181
 idle, 180
 naming, 180
 performance, 179
 thread starvation, 179
 timeouts, 180
 work queues, 180
 threads
 removing, 180
 See thread pools, 179
 timeouts, 86, 87
 thread pools, 180
 timer service
 See Enterprise JavaBeans
 timer service, 87-88
 timers
 See Enterprise JavaBeans
 timer service, 87-88
 topics, JMS, 55-57
 total-beans-created, 199
 total-beans-destroyed, 199
 total-num-errors, 197
 total-num-success, 197
 transaction management, 33
 Transaction Manager
 See transactions
 managers, 156
 transaction service, monitoring, 207
 transactions, 155
 associating, 156

transactions (*Continued*)
 attributes, 156
 committing, 156
 completing, 156
 demarcations, 156
 distributed, 156
 Enterprise JavaBeans, 85
 managers, 156
 recovering, 156
 rolling back, 156
 truststore.jks file, 127-128

U

UDDI, 166
 Universal Description, Discovery, and Integration
 (UDDI), 166

V

virtual servers, overview, 161-162

W

Web services, registries, 170
 web services, 33
 Web services
 as JBI providers, 176
 deploying, 168
 management, 165
 monitoring, 174
 monitoring statistics for, 174
 publishing, 170, 173
 security, 169
 testing, 169
 transforming with XSLT, 175
 viewing attributes of, 168
 viewing messages for, 175
 web sessions, See HTTP sessions, 83
 work queues, See thread pools, 180

X

XSLT, using, 175