



Sun GlassFish Communications Server 1.5 Administration Guide



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

Part No: 820-4281-11
October 2009

Copyright 2009 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. or its subsidiaries 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 Sun™ 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 2009 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., ou ses filiales, 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 CONTREFAÇON.

Contents

Preface	21
1 Communications Server Overview	25
Communications Server Overview and Concepts	25
Communications Server Overview	25
Tools for Administration	26
Communications Server Concepts	28
Domain	28
Domain Administration Server (DAS)	29
Usage Profiles	29
Cluster	30
Node Agent	31
Server Instance	31
Basic Communications Server Commands	33
Creating a Domain	34
Deleting a Domain	35
Listing Domains	35
Starting the Domain	35
Starting the Default Domain on Windows	35
Stopping the Domain	35
Stopping the Default Domain on Windows	36
Restarting the Domain	36
Creating a Cluster	36
Starting a Cluster	36
Stopping a Cluster	37
Creating a Node Agent	37
Starting a Node Agent	37
Stopping a Node Agent	37

Starting an Instance	38
Stopping an Instance	38
Restarting an Instance	38
Recreating the Domain Administration Server	38
2 Java Business Integration	41
JBI Environment	41
JBI Components	41
Service Assemblies	43
Shared Libraries	44
JBI Descriptors	44
3 JDBC Resources	45
JDBC Resources	45
JDBC Connection Pools	46
How JDBC Resources and Connection Pools Work Together	46
Setting Up Database Access	47
Working with JDBC Connection Pools	48
Creating a JDBC Connection Pool	48
Editing a JDBC Connection Pool	50
Editing JDBC Connection Pool Advanced Attributes	52
Configurations for Specific JDBC Drivers	54
Java DB Type 4 Driver	55
Sun GlassFish JDBC Driver for DB2 Databases	56
Sun GlassFish JDBC Driver for Oracle 8.1.7 and 9.x Databases	56
Sun GlassFish JDBC Driver for Microsoft SQL Server Databases	57
Sun GlassFish JDBC Driver for Sybase Databases	57
IBM DB2 8.1 Type 2 Driver	58
JConnect Type 4 Driver for Sybase ASE 12.5 Databases	58
MySQL Type 4 Driver	59
Inet Oraxo JDBC Driver for Oracle 8.1.7 and 9.x Databases	59
Inet Merlia JDBC Driver for Microsoft SQL Server Databases	60
Inet Sybelux JDBC Driver for Sybase Databases	61
Oracle Thin Type 4 Driver for Oracle 8.1.7 and 9.x Databases	61
OCI Oracle Type 2 Driver for Oracle 8.1.7 and 9.x Databases	62

IBM Informix Type 4 Driver	63
CloudScape 5.1 Type 4 Driver	63
4 Configuring Java Message Service Resources	65
JMS Resources	65
The Relationship Between JMS Resources and Connector Resources	66
JMS Connection Factories	67
JMS Destination Resources	67
JMS Physical Destinations	68
Configuring JMS Provider Properties	68
Accessing Remote Servers	69
Foreign JMS Providers	69
Configuring the Generic Resource Adapter for JMS	69
Resource Adapter Properties	70
ManagedConnectionFactory Properties	73
Administered Object Resource Properties	74
Activation Spec Properties	74
5 Configuring JavaMail Resources	77
Creating a JavaMail Session	77
6 JNDI Resources	79
Java EE Naming Services	79
Naming References and Binding Information	80
Using Custom Resources	81
Using External JNDI Repositories and Resources	81
7 Connector Resources	83
An Overview of Connectors	83
Managing Connector Connection Pools	84
▼ To Create a Connector Connection Pool	84
▼ To Edit a Connector Connection Pool	85
▼ To Edit Connector Connection Pool Advanced Attributes	87
▼ To Edit Connection Pool Properties	89

Managing Security Maps	89
▼ To Delete a Connector Connection Pool	89
▼ To Set Up EIS Access	90
Managing Connector Resources	90
▼ To Create a Connector Resource	90
▼ To Edit a Connector Resource	91
▼ To Delete a Connector Resource	91
▼ To Configure the Connector Service	92
Managing Administered Object Resources	92
▼ To Create an Administered Object Resource	93
▼ To Edit an Administered Object Resource	94
▼ To Delete an Administered Object Resource	94
8 SIP, Web, and EJB Containers	95
The SIP Servlet Container	95
Editing the Properties of the SIP Container	96
The Web Container	98
The EJB Container	98
9 Configuring Security	99
Understanding Application and System Security	99
Tools for Managing Security	100
Managing Security of Passwords	101
Encrypting a Password in the domain.xml File	101
Protecting Files with Encoded Passwords	102
Changing the Master Password	102
Working with the Master Password and Keystores	103
Changing the Admin Password	103
About Authentication and Authorization	104
Authenticating Entities	104
Authorizing Users	105
Specifying JACC Providers	105
Auditing Authentication and Authorization Decisions	105
Configuring Message Security	106
Understanding Users, Groups, Roles, and Realms	106

Users	107
Groups	107
Roles	108
Realms	108
Trust Configurations and Entities	110
Creating Trust Configurations and Entities	110
Editing Trust Configurations and Entities	110
Introduction to Certificates and SSL	111
About Digital Certificates	111
About Secure Sockets Layer	113
About Firewalls	114
About Certificate Files	114
Changing the Location of Certificate Files	115
Using Java Secure Socket Extension (JSSE) Tools	115
Using the keytool Utility	116
Generating a Certificate Using the keytool Utility	117
Signing a Digital Certificate Using the keytool Utility	118
Deleting a Certificate Using the keytool Utility	119
10 Configuring Message Security	121
Overview of Message Security	121
Understanding Message Security in the Communications Server	122
Assigning Message Security Responsibilities	122
About Security Tokens and Security Mechanisms	123
Glossary of Message Security Terminology	125
Securing a Web Service	126
Configuring Application-Specific Web Services Security	127
Securing the Sample Application	127
Configuring the Communications Server for Message Security	127
Actions of Request and Response Policy Configurations	128
Configuring Other Security Facilities	129
Configuring a JCE Provider	129
Message Security Setup	131
Enabling Providers for Message Security	131
Configuring the Message Security Provider	132

Creating a Message Security Provider	133
Enabling Message Security for Application Clients	133
Setting the Request and Response Policy for the Application Client Configuration	133
Further Information	134
11 Configuring the Diagnostic Service	135
What is the Diagnostic Framework?	135
Diagnostic Service Framework	135
Generating a Diagnostic Report	136
12 Transactions	137
About Transactions	137
What is a Transaction?	137
Transactions in Java EE Technology	138
Workarounds for Specific Databases	139
Admin Console Tasks for Transactions	139
Configuring Transactions	139
13 Configuring the HTTP Service	143
Virtual Servers	143
HTTP Listeners	144
14 Configuring the SIP Service	147
SIP Service	147
Setting the SIP Service Access Log Attributes	147
Setting the SIP Service Keep Alive Attributes	149
Setting the SIP Service Connection Pool Attributes	149
Setting the SIP Service Protocol Attributes	149
SIP Listeners	150
Creating and Editing a SIP Listener	151
▼ To Configure a SIP Listener to Use SSL	151
15 Overload Protection	153
CPU Overload Protection	153

Memory Overload Protection	154
16 Managing Web Services	155
Overview of Web Services	155
Web Services Standards	156
Java EE Web Service Standards	156
Deploying and Testing Web Services	157
Deploying Web Services	157
Viewing Deployed Web Services	158
Testing Web Services	158
Web Services Security	158
Using Web Services Registries	158
Adding a Registry	159
Publishing a Web Service to a Registry	159
Transforming Messages with XSLT Filters	160
Monitoring Web Services	160
Viewing Web Service Statistics	161
Monitoring Web Service Messages	161
17 Configuring the Object Request Broker	163
An Overview of the Object Request Broker	163
CORBA	163
What is the ORB?	164
IIOP Listeners	164
Configuring the ORB	164
Managing IIOP Listeners	164
18 Thread Pools	165
Working with Thread Pools	166
Configuring ThreadPools for SIP Container	166
19 Configuring Logging	169
About Logging	169
Log Records	169

The Logger Namespace Hierarchy	170
Configuring Logging	172
Configuring General Logging Settings	172
Configuring Log Levels	172
Viewing Server Logs	173
20 Monitoring Components and Services	175
About Monitoring	175
Monitoring in the Communications Server	175
Overview of Monitoring	176
The Tree Structure of Monitorable Objects	176
Statistics for Monitored Components and Services	180
Enabling and Disabling Monitoring	198
Configuring Monitoring Levels Using the Admin Console	199
▼ To Configure Monitoring Levels Using asadmin	199
Viewing Monitoring Data	200
Viewing Monitoring Data in the Admin Console	200
Viewing Monitoring Data With the asadmin Tool	200
Using JConsole	216
Securing JConsole to Application Server Connection	217
Prerequisites for Connecting JConsole to Application Server	218
▼ Connecting JConsole to Application Server	218
▼ Connecting JConsole Securely to Application Server	219
21 Configuring Management Rules	221
About Management Rules	221
Configuring Management Rules	222
22 Java Virtual Machine and Advanced Settings	225
Tuning the JVM Settings	225
Configuring Advanced Settings	226
A Automatically Restarting a Domain or Node Agent	227
Restarting Automatically on Solaris 10	227

Restarting Automatically Using inittab on Solaris 9 and Linux Platforms	229
Restarting Automatically on the Microsoft Windows Platform	229
Creating a Windows Service	229
Preventing the Service From Shutting Down When a User Logs Out	231
Security for Automatic Restarts	231
B Dotted Name Attributes for domain.xml	233
Top Level Elements	233
Elements Not Aliased	235
C The asadmin Utility	237
The asadmin Utility	238
Common Options for Remote Commands	240
The Multimode Command	241
The Get, Set, and List Commands	242
Server Lifecycle Commands	243
List and Status Commands	244
Deployment Commands	245
Version Commands	246
Message Queue Administration Commands	246
Resource Management Commands	247
Configuration Commands	249
SIP Listener Commands	249
HTTP and IIOP Listener Commands	249
Lifecycle and Audit Module Commands	250
Profiler and SSL Commands	250
JVM Options and Virtual Server Commands	251
Threadpool and Auth-Realm Commands	251
Transaction and Timer Commands	252
Registry Commands	252
User Management Commands	253
Rules and Monitoring Commands	253
Database Commands	254
Diagnostic and Logging Commands	254
Web Service Commands	255

Security Service Commands	255
Password Commands	256
Verify Command	257
Custom MBean Commands	257
Service Command	258
Property Command	258
Index	259

Figures

FIGURE 1-1	Communications Server Instance	32
FIGURE 9-1	Role Mapping	107

Tables

TABLE 1-1	Features Available for Each Profile	30
TABLE 1-2	Communications Server Listeners that Use Ports	33
TABLE 6-1	JNDI Lookups and Their Associated References	81
TABLE 9-1	Communications Server Authentication Methods	104
TABLE 10-1	Message protection policy to WS-Security SOAP message security operation mapping	128
TABLE 19-1	Communications Server Logger Namespaces	170
TABLE 20-1	SIP Service Network Manager Statistics	181
TABLE 20-2	Overload Protection Manager Statistics	181
TABLE 20-3	Session Manager Statistics	181
TABLE 20-4	TransactionManager Statisticsa	182
TABLE 20-5	EJB Statistics	183
TABLE 20-6	EJB Method Statistics	183
TABLE 20-7	EJB Session Store Statistics	184
TABLE 20-8	EJB Pool Statistics	185
TABLE 20-9	EJB Cache Statistics	186
TABLE 20-10	Timer Statistics	187
TABLE 20-11	Web Container (Servlet) Statistics	187
TABLE 20-12	Web Container (Web Module) Statistics	187
TABLE 20-13	HTTP Service Statistics (Developer Profile)	189
TABLE 20-14	JDBC Connection Pool Statistics	190
TABLE 20-15	Connector Connection Pool Statistics	191
TABLE 20-16	Connector Work Management Statistics	192
TABLE 20-17	Connection Manager (in an ORB) Statistics	192
TABLE 20-18	Thread Pool Statistics	193
TABLE 20-19	Transaction Service Statistics	193
TABLE 20-20	JVM Statistics	194
TABLE 20-21	JVM Statistics for Java SE- Class Loading	194
TABLE 20-22	JVM Statistics for Java SE- Compilation	195

TABLE 20-23	JVM Statistics for Java SE- Garbage Collection	195
TABLE 20-24	JVM Statistics for Java SE- Memory	195
TABLE 20-25	JVM Statistics for Java SE - Operating System	196
TABLE 20-26	JVM Statistics for Java SE - Runtime	196
TABLE 20-27	JVM Statistics for Java SE - Thread Info	197
TABLE 20-28	JVM Statistics for Java SE - Threads	198
TABLE 20-29	Top Level	210
TABLE 20-30	Applications Level	210
TABLE 20-31	Applications - Enterprise Applications and Standalone Modules	211
TABLE 20-32	HTTP-Service Level	214
TABLE 20-33	Thread-Pools Level	214
TABLE 20-34	Resources Level	215
TABLE 20-35	Transaction-Service Level	215
TABLE 20-36	ORB Level	215
TABLE 20-37	JVM Level	216
TABLE C-1	Remote Commands Required Options	240
TABLE C-2	Server Lifecycle Commands	243
TABLE C-3	List and Status Commands	244
TABLE C-4	Deployment Commands	245
TABLE C-5	Version Commands	246
TABLE C-6	Message Queue Commands	246
TABLE C-7	Resource Management Commands	247
TABLE C-8	SIP Listener Commands	249
TABLE C-9	IIOP Listener Commands	249
TABLE C-10	Lifecycle Module Commands	250
TABLE C-11	Profiler and SSL Commands	250
TABLE C-12	JVM Options and Virtual Server Commands	251
TABLE C-13	Threadpool and Auth-Realm Commands	252
TABLE C-14	Transaction Commands	252
TABLE C-15	Transaction Commands	253
TABLE C-16	User Management Commands	253
TABLE C-17	Rules and Monitoring Commands	253
TABLE C-18	Database Commands	254
TABLE C-19	Diagnostic and Logging Commands	254
TABLE C-20	Web Service Commands	255
TABLE C-21	Security Commands	256

TABLE C-22	Password Commands	257
TABLE C-23	Verify Command	257
TABLE C-24	Custom MBean Commands	257
TABLE C-25	Service Command	258
TABLE C-26	Property Command	258

Examples

EXAMPLE 20-1	Applications Node Tree Structure	177
EXAMPLE 20-2	SIP Service Tree Structure	178
EXAMPLE 20-3	HTTP Service Schematic (Developer Profile Version)	178
EXAMPLE 20-4	HTTP Service Schematic (Cluster and Enterprise Profile Version)	178
EXAMPLE 20-5	Resources Schematic	178
EXAMPLE 20-6	Connector Service Schematic	179
EXAMPLE 20-7	JMS Service Schematic	179
EXAMPLE 20-8	ORB Schematic	179
EXAMPLE 20-9	Thread Pool Schematic	180
EXAMPLE C-1	Passwordfile contents	239

Preface

The Administration guide describes the administrative tasks of the Communications Server.

This preface contains information about and conventions for the entire Sun GlassFish™ Communications Server documentation set.

Sun GlassFish Communications Server Documentation Set

TABLE P-1 Books in the Communications Server Documentation Set

Book Title	Description
<i>Documentation Center</i>	Communications 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 Communications 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 Communications 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 Communications 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>Java WSIT Tutorial</i>	Developing web applications using the Web Service Interoperability Technologies (WSIT). Describes how, when, and why to use the WSIT technologies and the features and options that each technology supports.
<i>Administration Guide</i>	System administration for the Communications Server, including configuration, monitoring, security, resource management, and web services management.

TABLE P-1 Books in the Communications Server Documentation Set (Continued)

Book Title	Description
<i>High Availability Administration Guide</i>	Setting up clusters, working with node agents, and using load balancers.
<i>Administration Reference</i>	Editing the Communications Server configuration file, <code>domain.xml</code> .
<i>Performance Tuning Guide</i>	Tuning the Communications Server to improve performance.
<i>Reference Manual</i>	Utility commands available with the Communications Server; written in man page style. Includes the <code>asadmin</code> command line interface.

Default Paths and File Names

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

TABLE P-2 Default Paths and File Names

Placeholder	Description	Default Value
<i>as-install</i>	Represents the base installation directory for Communications Server.	Solaris™ and Linux 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.	All installations: <i>as-install/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>
<i>instance-dir</i>	Represents the directory for a server instance.	<i>domain-dir/instance-dir</i>
<i>samples-dir</i>	Represents the directory containing sample applications.	<i>as-install/samples</i>
<i>docs-dir</i>	Represents the directory containing documentation.	<i>as-install/docs</i>

Typographic Conventions

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

TABLE P-3 Typographic Conventions

Typeface	Meaning	Example
AaBbCc123	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> . <i>A 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-4 Symbol Conventions

Symbol	Description	Example	Meaning
[]	Contains optional arguments and command options.	<code>ls [-l]</code>	The <code>-l</code> option is not required.
{ }	Contains a set of choices for a required command option.	<code>-d {y n}</code>	The <code>-d</code> option requires that you use either the <code>y</code> argument or the <code>n</code> argument.
`\${ }`	Indicates a variable reference.	<code>\${com.sun.javaRoot}</code>	References the value of the <code>com.sun.javaRoot</code> 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.

TABLE P-4 Symbol Conventions (Continued)

Symbol	Description	Example	Meaning
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (<http://www.sun.com/documentation/>)
- Support (<http://www.sun.com/support/>)
- Training (<http://www.sun.com/training/>)

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 Feedback. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document.

Communications Server Overview

Sun GlassFish Communications Server administration includes many tasks such as deploying converged and SIP applications, creating and configuring domains, server instances and resources; controlling (starting and stopping) domains and server instances, managing profiles and clusters, monitoring and managing performance, and diagnosing and troubleshooting problems.

Communications Server adds capabilities related to Session Initiation Protocol (SIP) on top of GlassFish.

Communications Server provides a Java EE compatible server for the development and deployment of Java EE converged, web, and SIP applications and Java Web Services. Key features include scalable transaction management, container-managed persistence runtime, performant web services, clustering, high availability, security, and integration capabilities.

Communications Server Overview and Concepts

This section contains the following topics:

- [“Communications Server Overview” on page 25](#)
- [“Tools for Administration” on page 26](#)

Communications Server Overview

Communications Server supports the Java EE 5 specification and the SIP Servlets 1.1 API. Communications Server based on the Sun GlassFish Enterprise Server 2.1. Communications Server is built on top of the core architecture and administrative infrastructure of GlassFish. Communications Server blends SIP Servlets technology with technologies such as EJB, Java Persistence API, JDBC, Web services, and reuse container services like JNDI, JMS, Dependency Injection, Security, and Transaction management.

Communications Server SIP protocol stack implemented using Grizzly (<http://grizzly.dev.java.net>). Incoming requests pass through a chain of handlers that perform specific functions. For example, the Overload Handler, if enabled, would send an appropriate error response, if the server is overloaded. The optional load balancer, if enabled, would apply the specified sticky load balancing logic and forward the request to the appropriate instance in the cluster, or pass it up the stack. Initial requests are matched with deployed applications and appropriately routed. Optionally, the default or user deployed Application Router is consulted to make the initial routing decisions. Many SIP listeners can be configured to accept TCP and or UDP traffic on multiple network interfaces. SIP over TLS is supported.

You can configure and monitor Communications Server instances and clusters securely and remotely, from a web-based central administration console. Administration in a carrier deployment environment can be automated in many ways. The command line interface (CLI) can be used to script and automate processes. A stable JMX API is made available to programmatically monitor the server, query configuration and change configuration data. Monitoring is supported through JMX and SNMP interfaces. Monitoring level may be varied dynamically from OFF to LOW and high, changing the amount of information that is collected.

You can configure and optimize Communications Server for a particular type of use by specifying the appropriate profile when you create the application server domain.

Communications Server includes the Metro web services stack (<http://metro.dev.java.net>). Metro implements important WS-* standards and WS-I standardized interoperability profiles in order to assure interoperability between Java and .NET web services.

Communications Server includes the implementation of Java Business Integration (JBI) specifications. JBI is a Java standard for structuring business systems according to a Service-Oriented Architecture (SOA).

Tools for Administration

The Communications Server provides the following administration tools and APIs:

- “Admin Console” on page 26
- “Command-line Interface (asadmin Utility)” on page 27
- “JConsole” on page 28

Admin Console

The Admin Console is a browser-based tool that features an easy-to-navigate interface and online help. The administration server (also called the Domain Administration Server or DAS) must be running to use the Admin Console. To launch the Administration Console, you must know the administration server hostname and port number. When the Communications 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 Communications Server was installed, specify `localhost` for the host name.

On Windows, start the Communications 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.

If your configuration includes remote server instances, create node agents to manage and facilitate remote server instances. It is the responsibility of the node agent to create, start, stop, and delete a server instance. Use the command line interface (CLI) commands to set up node agents.

Command-line Interface (`asadmin` Utility)

The `asadmin` utility is a command-line interface for the Sun GlassFish Communications Server. Use the `asadmin` utility and the commands associated with it to perform the same set of administrative tasks offered by the Admin Console. The default installation root directory on Solaris is `/opt/SUNWappserver`.

To start the `asadmin` utility, go to the `as-install/bin` directory and enter:

```
$ ./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 and PDF format in the [Sun GlassFish Communications Server 1.5 Reference Manual](#).

JConsole

JConsole is used to monitor the Sun GlassFish Communications Server. You can use either the JConsole remote tab, or the advanced tab to connect to the Communications Server.

- Remote Tab: identify the username, password, administration server host, and JMS port number (8686 by default), and select Connect.
- Advanced Tab: identify the `JMXServiceURL` as `service:jmx:rmi:///jndi/rmi://host:jms-port/jmxrmi` and select Connect. The `JMXServerURL` is printed in the `server.log` file as well as output in the command window of the domain creation command.

Communications Server Concepts

The Communications Server consists of one or more domains. A domain is an administrative boundary or context. Each domain has an administration server (also called Domain Administration Server or DAS) associated with it and consists of zero or more standalone instances and/or clusters. Each cluster has one or more homogeneous server instances. A server instance is a single Java Virtual Machine (JVM) that runs the Application Server on a single physical machine. Server instances (whether standalone or clustered) in a domain can run on different physical hosts.

This section contains the following topics:

- “Domain” on page 28
- “Domain Administration Server (DAS)” on page 29
- “Usage Profiles” on page 29
- “Cluster” on page 30
- “Node Agent” on page 31
- “Server Instance” on page 31

Domain

A domain is a group of instances that are administered together. However, an application server instance can belong to just one domain. In addition to the administration boundary, a domain provides the basic security structure whereby different administrators can administer specific groups (domains) of application server instances. By grouping the server instances into separate domains, different organizations and administrators can share a single Communications Server installation. Each domain has its own configuration, log files, and application deployment areas

that are independent of other domains. If the configuration is changed for one domain, the configurations of other domains are not affected.

The Sun GlassFish Communications Server installer creates the default administrative domain (named `domain1`). It also creates an associated domain administration server (named `server`). You must provide the administration server port number. The default administration server port is 4848. The installer also queries for the administration username and master password. After installation, additional administration domains can be created.

Domain Administration Server (DAS)

Each domain has its own Domain Administration Server (DAS) with a unique port number. The Admin Console communicates with a specific DAS to administer the associated domain. Each Admin Console session allows you to configure and manage the specific domain.

The Domain Administration Server (DAS), is a specially-designated application server instance that hosts the administrative applications. The DAS authenticates the administrator, accepts requests from administration tools, and communicates with server instances in the domain to carry out the requests. The DAS is sometimes referred to as the admin server or default server. It is referred to as the default server because it is the only server instance that gets created on Sun GlassFish Communications Server installation and can be used for deployments. The DAS is simply a server instance with additional administration capabilities.

Each Admin Console session allows you to configure and manage a single domain. If you created multiple domains, you must start an additional Admin Console session to manage the other domains. When specifying the URL for the Admin Console, be sure to use the port number of the DAS associated with the domain to be administered.

Usage Profiles

Every administrative domain is associated with a usage profile, which identifies the capabilities of that domain. Communications Server provides the following profiles:

- **Developer:** Use this profile if you are running your domain in a development environment and if your applications do not need the NSS keystore or clustering features, such as load balancing, and session persistence.
- **Cluster:** Use this profile if you need to create clusters but do not require the high-availability database (HADB) or the NSS keystore.

Note – In the cluster and developer profile, the DAS is set to be in non-secure mode, by default. While running `asadmin` commands, use `--secure=false`, wherever required. Do not use this flag if you have made changes to set DAS to secure mode.

The domain provides a preconfigured runtime for the user applications. Usage profiles facilitates the distinction between the Application Server binaries and the runtime configuration. Profiles enable you to use the same installation of Communications Server to create different domains with profiles that suit specific needs. For example, a developer may want to use the Communications Server to get to know the latest Java EE specifications. This developer does not need stringent security settings. Another user who wants to deploy applications in a production environment needs an inherently secure environment.

Table 1-1 lists the features available with each profile:

TABLE 1-1 Features Available for Each Profile

Feature	Developer Profile	Cluster Profile	Enterprise Profile (not available with Sun GlassFish Communications Server)
Security store	JKS	JKS	NSS
Clustering/Standalone instances	Not available	Available	Available
Security Manager	Disabled	Enabled	Enabled
HADB	Not available	Not available	Available
Load balancing	Not available	Available	Available
Node agents	Not available	Available	Available

Cluster

A cluster is a named collection of server instances sharing the same set of applications, resources, and configuration information. A server instance can belong to exactly one cluster. A cluster facilitates server instance load-balancing through distribution of a load across multiple machines. A cluster facilitates high availability through instance-level failover. From an administrative perspective, a cluster represents a virtualized entity in which operations on a cluster (e.g. deployment of an application) act on all instances that make up the cluster.

Horizontal scaling is achieved by adding Communications Server instances to a cluster, thereby increasing the capacity of the system. It is possible to add Communications Server instances to a cluster without disrupting service. The HTTP, RMI/IIOP, and JMS load balancing systems distribute requests to healthy Communications Server instances in the cluster.

High Availability - Availability allows for failover protection of Communications Server instances in a cluster. If one application server instance goes down, another Communications Server instance takes over the sessions that were assigned to the unavailable server. Session information is stored using the session replication feature or by using the high-availability database (HADB). HADB supports the persistence of HTTP sessions and stateful session beans.

Node Agent

A lightweight agent (e.g. hosting a JMX runtime only) is required on each node in the domain to facilitate remote lifecycle management of instances. Its primary purpose is to start, stop, and create server instances as instructed by the DAS. The Node Agent also acts as a watchdog and restarts failed processes. Like the DAS, the Node Agent should only be required for certain administrative operations and should not be expected to be highly available. However, the Node Agent is an “always on” component, and must be configured to be started by the native O/S node bootstrap (e.g. Solaris/Linux `inetd`, or as a Windows service). A Node Agent is not required for the DAS.

Server Instance

The server instance is a single Java EE compatible Java Virtual Machine hosting an Communications Server on a single node. Each server instance has a unique name in the domain. A clustered server instance is a member of a cluster and receives all of its applications, resources, and configuration from its parent cluster; ensuring that all instances in the cluster are homogeneous. An unclustered server instance does not belong to a cluster and as such has an independent set of applications, resources, and configuration. The following figure shows an application server instance in detail. The application server instance is a building block in the clustering, load balancing, and session persistence features of the Communications Server.

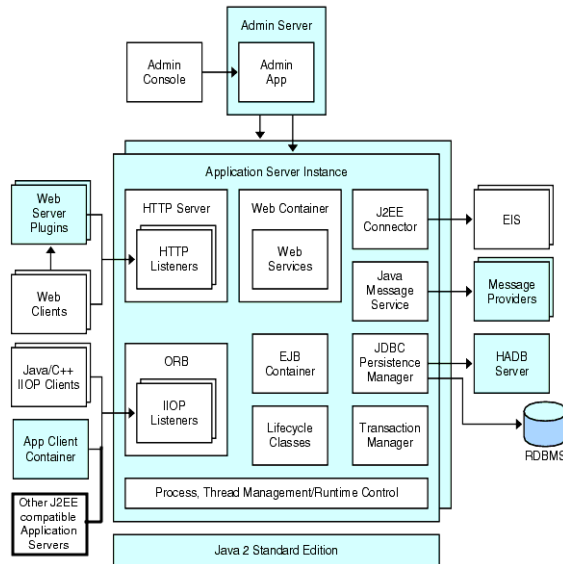


FIGURE 1-1 Communications Server Instance

The Sun GlassFish Communications Server creates one application server instance, called server, at the time of installation. For many users, one application server instance meets their needs. However, depending upon your environment, you might want to create one or more additional application server instances. For example, in a development environment you can use different application server instances to test different Communications Server configurations, or to compare and test different application deployments. Because you can easily add or delete an application server instance, you can use them to create temporary sandbox area for experimentation purposes.

In addition, for each application server instance, you can also create virtual servers. Within a single installed application server instance you can offer companies or individuals domain names, IP Addresses, and some administration capabilities. For the users, it is almost as if they have their own web server, without the hardware and basic server maintenance. These virtual servers do not span application server instances. For more information about virtual servers, see [Chapter 13, “Configuring the HTTP Service.”](#)

In operational deployments, for many purposes you can use virtual servers instead of multiple application server instances. However, if virtual servers do not meet your needs, you can also use multiple application server instances. On stopping, application server instance stops accepting new connections, then waits for all outstanding connections to complete. If your machine crashes or is taken offline, the server quits and any requests it was servicing may be lost.

Ports in the Communications Server

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

TABLE 1-2 Communications 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 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	3700	Remote clients of enterprise beans (EJB components) access the beans through the IIOP listener.
IIOP, SSL	3820	Another port is used by the IIOP listener configured for secure communications.
IIOP, SSL and mutual authentication	3920	Another port is used by the IIOP listener configured for mutual (client and server) authentication.
SIP	5060	The server listens for SIP requests on a port.
SIPS	5061	SIP/converged applications configured for secure communications listen on a separate port.
JMX_ADMIN	8686	
JMS	7676	

Basic Communications Server Commands

Administration of the Communications Server includes tasks such as creation, configuration, control and management of domains, clusters, node agents, and server instances. This section contains the following topics:

- “Creating a Domain” on page 34
- “Deleting a Domain” on page 35
- “Listing Domains” on page 35
- “Starting the Domain” on page 35
- “Starting the Default Domain on Windows” on page 35
- “Stopping the Domain” on page 35
- “Stopping the Default Domain on Windows” on page 36
- “Restarting the Domain” on page 36
- “Creating a Cluster” on page 36
- “Starting a Cluster” on page 36

- “Stopping a Cluster” on page 37
- “Creating a Node Agent” on page 37
- “Starting a Node Agent” on page 37
- “Stopping a Node Agent” on page 37
- “Starting an Instance” on page 38
- “Stopping an Instance” on page 38
- “Restarting an Instance” on page 38
- “Recreating the Domain Administration Server” on page 38

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 5000 and the administrative user name is `admin`. The command prompts for the administrative and master passwords.

```
$ asadmin create-domain --adminport 5000 --adminuser admin mydomain
```

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

```
http://hostname:5000
```

In Communications Server 9.1, every domain has a profile associated with it. For information on profiles, see “Usage Profiles” on page 29. You can choose the profile of a domain only during creation. Use the `--profile` option with the `create-domain` command to specify a profile for the domain. If you do not use the `--profile` option to explicitly specify a profile, the default profile is associated with the domain. The `AS_ADMIN_PROFILE` variable in the `asadminenv.conf` file defines the default profile.



Caution – Do not create an enterprise domain unless you have HADB and the Network Security Services (NSS) keystore. You will not be able to start an enterprise domain unless you have HADB and NSS.

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` or the `create-domain(1)`.

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. Each domain must be started separately.

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
```

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

Starting the Default Domain on Windows

From the Windows Start Menu, select Programs -> Sun Microsystems -> Communications 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`.

Consult the Admin Console online help to stop the domain through the Admin Console.

Stopping the Default Domain on Windows

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

Restarting the Domain

Restarting the server is the same as restarting the domain. To restart the domain or server, stop and start the domain.

Creating a Cluster

A cluster is created using the `create-cluster` command. The following example creates a cluster named `mycluster`. The administration server host is `myhost`, the server port is `1234`, and the administrative username is `admin`. The command prompts for the administrative passwords.

```
$ asadmin create-cluster --host myhost --port 1234 --user admin mycluster
```

For the full syntax, type `asadmin help create-cluster`.

Starting a Cluster

A cluster is started using the `start-cluster` command. The following example starts the cluster named `mycluster`. The command prompts for the administrative passwords.

```
$ asadmin start-cluster --host myhost --port 1234 --user admin mycluster
```

For the full syntax, type `asadmin help start-cluster`.

Stopping a Cluster

A cluster is stopped using the `stop-cluster` command. The following example stops the cluster named `mycluster`. The command prompts for the administrative passwords.

```
$ asadmin stop-cluster --host myhost --port 1234 --user admin mycluster
```

`myhost` is the administrative server host, `1234` is the administrative port, `admin` is the administrative username.

For the full syntax, type `asadmin help stop-cluster`. When a cluster is stopped, all the server instances in the cluster get stopped. A cluster without instances cannot be stopped.

Creating a Node Agent

A node agent is created using the `create-node-agent` command. The following example creates node agent named `mynodeagent`. The administration server host is `myhost`, the administration server port is `1234`, and the administrative username is `admin`. The command normally prompts for the administrative passwords; however, if the `--savemasterpassword` option is not specified or `false`, the command does not prompt for the administrative passwords.

```
$ asadmin create-node-agent --host myhost --port 1234 --user admin mynodeagent
```

For the full syntax, type `asadmin help create-node-agent`.

Starting a Node Agent

A node agent is started using the `start-node-agent` command and specifying the node agent name. For example, to start the node agent `mynodeagent`, type the following:

```
$ asadmin start-node-agent --user admin mynodeagent
```

For the full syntax, type `asadmin help start-node-agent`.

Stopping a Node Agent

A node agent is stopped using the `stop-node-agent` command and specifying the node agent name. For example, to stop the node agent `mynodeagent`, type the following:

```
$ asadmin stop-node-agent mynodeagent
```

For the full syntax, type `asadmin help stop-node-agent`.

Starting an Instance

A server instance is started using the `start-instance` command. The following example starts the server instance named `myinstance`. The command prompts for the administrative passwords.

```
$ asadmin start-instance --host myhost --port 1234 --user admin myinstance
```

The administrative server host is `myhost`, the administrative port is `1234`, the administrative username is `admin`. The server instance `myinstance` can be clustered or standalone.

For the full syntax, type `asadmin help start-instance`.

Stopping an Instance

A server instance is started using the `stop-instance` command. The following example stops the server instance named `myinstance`. The command prompts for the administrative passwords.

```
$ asadmin stop-instance --host myhost --port 1234 --user admin myinstance
```

The administrative server host is `myhost`, the administrative port is `1234`, the administrative username is `admin`. The server instance `myinstance` can be clustered or standalone.

For the full syntax, type `asadmin help stop-instance`.

Restarting an Instance

To restart server instance, stop and start the instance.

Recreating the Domain Administration Server

For mirroring purposes, and to provide a working copy of the Domain Administration Server (DAS), you must have:

- One machine (`machine1`) that contains the original DAS.
- A second machine (`machine2`) that contains a cluster with server instances running applications and catering to clients. The cluster is configured using the DAS on the first machine.
- A third backup machine (`machine3`) where the DAS needs to be recreated in case the first machine crashes.

Note – You must maintain a backup of the DAS from the first machine. Use `asadmin backup-domain` to backup the current domain.

▼ To migrate the DAS

The following steps are required to migrate the Domain Administration Server from the first machine (machine1) to the third machine (machine3).

1 Install the application server on the third machine just as it is installed on the first machine.

This is required so that the DAS can be properly restored on the third machine and there are no path conflicts.

a. Install the application server administration package using the command-line (interactive) mode. To activate the interactive command-line mode, invoke the installation program using the `console` option:

```
./bundle-filename -console
```

You must have root permission to install using the command-line interface.

b. Deselect the option to install default domain.

Restoration of backed up domains is only supported on two machines with same architecture and **exactly** the same installation paths (use same *as-install* and *domain-root-dir* on both machines).

2 Copy the backup ZIP file from the first machine into the *domain-root-dir* on the third machine. You can also FTP the file.

3 Execute `asadmin restore-domain` command to restore the ZIP file onto the third machine:

```
asadmin restore-domain --filename domain-root-dir/sjsas_backup_v00001.zip domain1
```

You can backup any domain. However, while recreating the domain, the domain name should be same as the original.

4 Change *domain-root-dir/domain1/generated/tmp* directory permissions on the third machine to match the permissions of the same directory on first machine.

The default permissions of this directory are: `?drwx-----?` (or 700).

For example:

```
chmod 700 domain-root-dir/domain1/generated/tmp
```

The example above assumes you are backing up `domain1`. If you are backing up a domain by another name, you should replace `domain1` above with the name of the domain being backed up.

5 Change the host values for the properties in the `domain.xml` file for the third machine:**6 Update the `domain-root-dir/domain1/config/domain.xml` on the third machine.**

For example, search for `machine1` and replace it with `machine3`. So, you can change:

```
<jmx-connector><property name=client-hostname value=machine1/>...
```

to:

```
<jmx-connector><property name=client-hostname value=machine3/>...
```

7 Change:

```
<jms-service... host=machine1.../>
```

to:

```
<jms-service... host=machine3.../>
```

8 Start the restored domain on machine3:

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

9 Change the DAS host values for properties under node agent on machine2.**10 Change `agent.das.host` property value in**

`as-install/nodeagents/nodeagent/agent/config/das.properties` on machine2.

11 Restart the node agent on machine2.

Note – Start the cluster instances using the `asadmin start -instance` command to allow them to synchronize with the restored domain.

Java Business Integration

Java Business Integration (JBI) is an implementation of the [JSR 208 specification](http://www.jcp.org/en/jsr/detail?id=208) (<http://www.jcp.org/en/jsr/detail?id=208>) for Java Business Integration, a standard developed under the Java Community Process (JCP) as an approach to implementing a service-oriented architecture (SOA).

JBI defines an environment for plug-in components that interact using a services model based directly on Web Services Description Language (WSDL) 2.0. The plug-in components function as service providers, service consumers, or both.

For detailed information on managing the key components of the JBI runtime environment and their lifecycle states, see the Admin Console Online Help. For information about using the JBI commands, see *Sun GlassFish Communications Server 1.5 Reference Manual*.

JBI Environment

The key components of the JBI environment are covered in the following sections.

- “JBI Components” on page 41
- “Service Assemblies” on page 43
- “Shared Libraries” on page 44
- “JBI Descriptors” on page 44

JBI Components

Service Engines

Service Engines are components that provide local services (that is, services within the JBI environment) and consume local or remote services.

Binding Components

Binding Components are proxies for consumers or providers that are outside the JBI environment. Binding components typically are based on a standard communications protocol, such as FTP, JMS, or SMTP, or a call to an external service, such as SAP or WebSphere MQ.

JBI components have the following lifecycle states:

- Started
- Stopped
- Shutdown

The JBI Runtime persists the life cycle states of JBI Components. When the server shuts down and then restarts, JBI Components revert to their state at the time the Communications Server shut down.

Note – The JBI runtime attempts to revert to the "desired" state of a JBI component. For example, suppose you tried to start a JBI component but it did not start due to an error in the component. If you restart the Communications Server, the JBI runtime attempts to start the component again.

You can do the following operations on the JBI components. For detailed steps, log on to the Admin Console, navigate to the JBI node, click Components and then click Online Help.

- View JBI components by their specific lifestyle states.
- Install JBI components.
- Uninstall JBI components.
- Manage the lifecycle states of the JBI Components.
- View the general properties of a JBI Component.
- View the configuration information for a JBI Component.
- View the descriptor for a JBI Component.
- Manage JBI Component Loggers.

JBI Component Loggers

Using the Admin Console, you can manage the log levels for JBI Components. Some JBI Components provide several loggers while other components might not provide any. However, there will always be a logger level displayed for the entire component. But the logger level setting will only have an effect if a component implements its loggers based on the default name. The provider of a JBI Component might provide additional documentation on specifying logging levels.

Note – The logging levels for JBI Components are often inherited from a parent logger such as the JBI logger. To view and set parent logging levels, in the Admin Console, select Common Tasks and then Application Server. Then, in the Communications Server panel, select Logging and then Log Levels. Look for the drop-down list for the JBI module to view and set the parent JBI logging level.

Service Assemblies

A Service Assembly is a collection of Service Units that provision target components that together provide or consume specific services for an application. Service Assemblies are typically created in a development tools environment, such as that provided by NetBeans Enterprise Pack.

A Service Assembly has the following lifecycle states:

- Started
- Shutdown
- Stopped

The JBI Runtime persists the lifecycle states of Service Assemblies. When the Communications Server shuts down and then restarts, Service Assemblies revert to their state at the time the Communications Server shut down.

Note – The JBI runtime attempts to revert to the "desired" state of a Service Assembly. For example, suppose you tried to start a Service Assembly but it did not start due to an error in the Service Assembly. If you restart the Application Server, the JBI runtime attempts to start the Service Assembly again.

You can do the following operations on Service Assemblies. For detailed steps, log on to the Admin Console, navigate to the JBI node, click Service Assemblies and then click Online Help.

- View all the Service Assemblies, with support for sorting and for filtering by lifecycle state.
- Deploy a Service Assembly.
- Undeploy a Service Assembly.
- Manage the lifecycle of a Service Assembly.
- View the general properties of a Service Assembly.
- View the descriptor for a Service Assembly.

Shared Libraries

A Shared Library provides Java classes that are not private to a single component and is typically shared by more than one JBI Component. For example, the Java EE Service Engine requires the WSDL Shared Library.

You can do the following operations on Shared Libraries. For detailed steps, log on to the Admin Console, navigate to the JBI node, click Shared Libraries and then click Online Help.

- View all Shared Libraries.
- Install Shared Libraries.
- View General Properties of a Shared Library.
- View the descriptor for a Shared Library.
- Uninstall a Shared Library.

JBI Descriptors

The descriptor file (`jbi.xml`) for Service Assemblies, JBI Components, and Shared Libraries provides the following information:

- **Service Assemblies:** Lists the Service Units contained in the Service Assembly and the target for each Service Unit. Some Service Units may also show information on connection endpoints.
- **JBI Components:** Lists the type of the JBI Component (Binding Component or Service Engine), a description of the component, information about relevant classpaths for the component, and the name of any Shared Library upon which it depends.
- **Shared Libraries:** Lists the name of the shared library, and the names of the archive files (`.jar` files) or class file subdirectories it contains.

JDBC Resources

This chapter explains how to configure JDBC resources, which are required by applications that access databases. This chapter contains the following sections:

- “JDBC Resources” on page 45
- “JDBC Connection Pools” on page 46
- “How JDBC Resources and Connection Pools Work Together” on page 46
- “Setting Up Database Access” on page 47
- “Working with JDBC Connection Pools” on page 48
- “Configurations for Specific JDBC Drivers” on page 54

JDBC Resources

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

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.)

To create a JDBC resource, specify a unique JNDI name that identifies the resource. (See the section JNDI Names and Resources.) 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`.

A JDBC resource (data source) provides applications with a means of connecting to a database.

Before creating a JDBC resource, first create a JDBC connection pool.

When creating a JDBC resource, you must identify:

1. The JNDI Name. By convention, the name begins with the `jdbc/` string. For example: `jdbc/payrolldb`. Don't forget the forward slash.
2. Select a connection pool to be associated with the new JDBC resource.
3. Specify the settings for the resource.
4. Identify the targets (clusters and standalone server instance) on which the resource is available.

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.

The properties of connection pools can vary with different database vendors. Some common properties are the database's name (URL), user name, and password.

See Also:

- [“JDBC Resources” on page 45](#)
- [“How JDBC Resources and Connection Pools Work Together” on page 46](#)
- [“Editing a JDBC Connection Pool” on page 50](#)

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 is 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 is 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.

Setting Up Database Access

To set up a database access:

1. Install a supported database product.

For a list of database products supported by the Communications Server, see the *Release Notes*.

2. Install a JDBC driver for the database product.
3. Make the driver's JAR file accessible to the domain's server instance.
4. Create the database.

Usually, the application provider delivers scripts for creating and populating the database.

5. Create a connection pool for the database.
6. Create a JDBC resource that points to the connection pool.

Now to integrate the JDBC driver into an administrative domain, do either of the following:

1. Make the driver accessible to the common class loader.

Copy the driver's JAR and ZIP files into the *domain-dir/lib* directory or copy its class files into the *domain-dir/lib/ext* directory.

2. Restart the domain.
3. Identify the fully-qualified path name for the driver's JAR file.

Working with JDBC Connection Pools

A JDBC connection pool is a group of reusable connections for a particular database. When creating the pool with the Admin Console, the Administrator is actually defining the aspects of a connection to a specific database.

Before creating the pool, you must first install and integrate the JDBC driver. When building the Create Connection Pool pages, 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
- Required properties, such as the database name (URL), user name, and password

Creating a JDBC Connection Pool

You can use the Admin Console for creating the JDBC connection pool.

▼ Creating a JDBC Connection Pool and JDBC Resource Using the Admin Console

- 1 Copy the JDBC driver for database connectivity to `as-install/domains/domain_name/lib/ext`. Alternatively, add the path to the JDBC driver to `classpath-prefix` and restart the server.
- 2 In the Admin Console, click **Create JDBC Connection Pool** in the **Common Tasks** page.
- 3 Provide a name for the connection pool, select the resource type, select the database vendor. Click **Next**.
- 4 The Admin Console lists the default pool settings and the properties of `datasource` class provided by the vendor.
- 5 Click the **Ping** button to test the JDBC connection pool.
- 6 In the left-hand pane of the Admin Console, click **Resources > JDBC Resources**.
- 7 Click **New**.
- 8 In the **New JDBC Resource** page, provide the JNDI name for the resource and the JDBC connection pool with which the JDBC resource needs to be associated.

9 Click OK.

▼ Creating a JDBC Connection Pool and JDBC Resource Using the CLI

1 **Use the `asadmin create-jdbc-connection-pool` command to create a JDBC connection pool.**

Sample command to create a JDBC connection pool:

```
asadmin create-jdbc-connection-pool --datasourceclassname
oracle.jdbc.pool.OracleDataSource --restype javax.sql.DataSource --property
user=dbuser:password=dbpassword:url="jdbc:oracle:thin:@localhost:1521:ORCL"
myjdbc_oracle-pool
```

2 **Use the `asadmin create-jdbc-resource` command to create a JDBC resource associated with this connection pool.**

Sample command to create a JDBC resource:

```
asadmin create-jdbc-connection-pool --connectionpoolid myjdbc_oracle-pool
jdbc/oracle-resource
```

3 **Test the connection settings are correct by pinging the connection pool using the `asadmin ping-connection-pool` command.**

```
asadmin ping-connection-pool myjdbc_oracle-pool
```

4 **To edit JDBC connection pool properties, use the `asadmin list, get, and set` commands.**

Sample list command to list a JDBC connection pool:

```
asadmin list myjdbc_oracle-pool
```

The sample output of this command:

```
domain.resources.jdbc-connection-pool.oracle-pool
```

Sample get command to list all properties of a JDBC connection pool:

```
asadmin list myjdbc_oracle-pool
```

Sample get command to get all properties of a JDBC connection pool:

```
asadmin get domain.resources.jdbc-connection-pool.myjdbc_oracle-pool
```

Sample get command to set a properties of a JDBC connection pool:

```
asadmin set
domain.resources.jdbc-connection-pool.myjdbc_oracle-pool.steady-pool-size=20
```

Editing a JDBC Connection Pool

The Edit JDBC Connection Pool page provides the means to change all of the settings for an existing pool except its name.

1. Change general settings.

The values of the general settings depend on the specific JDBC driver that is installed. These settings are the names of classes or interfaces in the Java programming language.

Parameter	Description
DataSource Class Name	The vendor-specific class name that implements the <code>DataSource</code> and / or <code>XADataSource</code> APIs. This class is in the JDBC driver.
Resource Type	Choices include <code>javax.sql.DataSource</code> (local transactions only), <code>javax.sql.XADataSource</code> (global transactions), and <code>java.sql.ConnectionPoolDataSource</code> (local transactions, possible performance improvements).

2. Change pool settings.

A set of physical database connections reside in the pool. When an application requests a connection, the connection is removed from the pool, and when the application releases the connection, it is returned to the pool.

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.
Maximum Pool Size	The maximum number of connections in the pool.
Pool Resize Quantity	When the pool scales up and scales down towards the maximum and minimum pool sizes respectively, it is resized in batches. This value determines the number of connections in the batch. Making this value too large delays connection creation and 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 is removed from the pool.
Max Wait Time	The amount of time the application requesting a connection will wait before getting a connection timeout. Because the default wait time is long, the application might appear to hang indefinitely.

3. Change connection validation settings.

Optionally, the application server can validate connections before they are passed to applications. This validation allows the application server to automatically reestablish database connections if the database becomes unavailable due to network failure or database server crash. Validation of connections incurs additional overhead and slightly reduces performance.

Parameter	Description
Connection Validation	Select the Required checkbox to enable connection validation.
Validation Method	<p>The application server can validate database connections in three ways: auto-commit, metadata, and table.</p> <p>auto-commit and metadata - The application server validates a connection by calling the <code>con.setAutoCommit()</code> and <code>con.getMetaData()</code> methods.</p> <p>Auto-commit validation makes use of two methods for validating the connection. <code>con.setAutoCommit()</code> is used to retrieve the current state of auto-commit and <code>con.setAutoCommit()</code> to change the state of auto-commit. This allows actual contact with the database to take place.</p> <p><code>con.getAutoCommit()</code> might or might not contact the database, depending on the implementation. Actual physical connection will be wrapped for different purposes, such as a connection pool.</p> <p>Note – Because many JDBC drivers cache the results of these calls, they do not always provide reliable validations. Check with the driver vendor to determine whether these calls are cached or not.</p> <p>table - The application queries a database table that are specified. The table must exist and be accessible, but it doesn't require any rows. Do not use an existing table that has a large number of rows or a table that is already frequently accessed.</p>
Table Name	If you selected table from the Validation Method combo box, then specify the name of the database table here.
On Any Failure	If you select the checkbox labelled Close All Connections, if a single connection fails, then the application server closes all connections in the pool and then reestablish them. If you do not select the checkbox, then individual connections are reestablished only when they are used.
Allow Non Component Callers	Click this check box if you want to enable the pool for use by non-component callers such as Servlet Filters and Lifecycle modules.

4. Change transaction isolation settings.

Because a database is usually accessed by many users concurrently, one transaction might update data while another attempts to read the same data. The isolation level of a transaction defines the degree to which the data being updated is visible to other transactions. For details on isolation levels, refer to the documentation of the database vendor.

Parameter	Description
Non-transactional Connections	Click the check box if you want Application Server to return all non-transactional connections.
Transaction Isolation	Makes it possible to select the transaction isolation level for the connections of this pool. If left unspecified, the connections operate with default isolation levels provided by the JDBC driver.
Guaranteed Isolation Level	Only applicable if the isolation level has been specified. If you select the Guaranteed checkbox, then all connections taken from the pool have the same isolation level. For example, if the isolation level for the connection is changed programmatically (with <code>con.setTransactionIsolation</code>) when last used, this mechanism changes the status back to the specified isolation level.

5. Change properties.

In the Additional Properties table, it is possible to specify properties, such as the database name (URL), user name, and password. Because the properties vary with database vendor, consult the vendor's documentation for details.

Editing JDBC Connection Pool Advanced Attributes

To help diagnose connection leaks and improve ease-of-use, Application Server 9.1 provides several new attributes to configure a connection pool at the time of its creation.

1. Open the Advanced tab and specify the following attributes.

Attribute	Description
Name	Name of the JDBC connection pool whose properties you want to edit. You cannot change the pool name, however.
Statement Timeout	Time in seconds after which abnormally long running queries will be terminated. Application Server will set "QueryTimeout" on the statements created. The default value of -1 implies that the attribute is not enabled.
Wrap JDBC Objects	When set to true, application will get wrapped jdbc objects for Statement, PreparedStatement, CallableStatement, ResultSet, DatabaseMetaData. The default value is false.

2. Specify the Connection Settings as explained in the following table.

Attribute	Description
Validate Atmost Once	Amount of time, in seconds, after which a connection is validated at most once. This will help reduce the number of validation requests by a connection. The default value 0 implies that connection validation is not enabled.
Leak Timeout	Amount of time, in seconds, to trace connection leaks in a connection pool. The default value 0 means that connection leak tracing is disabled. If connection leak tracing is enabled, you can get statistics on the number of connection leaks in the Monitoring Resources tab. To view this tab, go to Application Server > Monitoring > Resources.
Leak Reclaim	If this option is enabled, leaked connections will be restored to the pool after leak connection tracing is complete.
Creation Retry Attempts	Number of attempts that will be made if there is a failure in creating a new connection. The default value of 0 implies that no attempts will be made to create the connection again.
Retry Interval	Specify the interval, in seconds, between two attempts to create a connection. The default value is 10 seconds. This attribute is used only if the value of Creation Retry Attempts is greater than 0.
Lazy Connection Enlistment	Enable this option to enlist a resource to the transaction only when it is actually used in a method.
Lazy Association	Connections are lazily associated when an operation is performed on them. Also, they are disassociated when the transaction is completed and a component method ends, which helps reuse of the physical connections. Default value is false.
Associate with Thread	Enable this option to associate a connection with the thread such that when the same thread is in need of a connection, it can reuse the connection already associated with that thread, thereby not incurring the overhead of getting a connection from the pool. Default value is false.

Match Connections	Use this option to switch on/off connection matching for the pool. It can be set to false if the administrator knows that the connections in the pool will always be homogeneous and hence a connection picked from the pool need not be matched by the resource adapter. Default value is false.
Max Connection Usage	Specify the number of times a connection should be reused by the pool. Once a connection is reused for the specified number of times, it will be closed. This is useful, for instance, to avoid statement-leaks. The default value of 0 implies that no connections will be reused.

Configurations for Specific JDBC Drivers

Communications 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 Communications Server and are found to be Java EE compatible.

- “Java DB Type 4 Driver” on page 55
- “Sun GlassFish JDBC Driver for DB2 Databases” on page 56
- “Sun GlassFish JDBC Driver for Oracle 8.1.7 and 9.x Databases” on page 56
- “Sun GlassFish JDBC Driver for Microsoft SQL Server Databases” on page 57
- “Sun GlassFish JDBC Driver for Sybase Databases” on page 57
- “IBM DB2 8.1 Type 2 Driver” on page 58
- “JConnect Type 4 Driver for Sybase ASE 12.5 Databases” on page 58
- “MySQL Type 4 Driver” on page 59

Other JDBC drivers can be used with Communications 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 Communications Server.

- “Inet Oraxo JDBC Driver for Oracle 8.1.7 and 9.x Databases” on page 59
- “Inet Merlia JDBC Driver for Microsoft SQL Server Databases” on page 60
- “Inet Sybelux JDBC Driver for Sybase Databases” on page 61
- “Oracle Thin Type 4 Driver for Oracle 8.1.7 and 9.x Databases” on page 61
- “OCI Oracle Type 2 Driver for Oracle 8.1.7 and 9.x Databases” on page 62
- “IBM Informix Type 4 Driver” on page 63
- “CloudScape 5.1 Type 4 Driver” on page 63

For details about how to integrate a JDBC driver and how to use the Administration Console or the command line interface to implement the configuration, see the [Sun GlassFish Communications Server 1.5 Administration Guide](#).

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 GlassFish Communications Server 1.5 Reference Manual](#).

Java DB Type 4 Driver

The Java DB JDBC driver is included with the Communications Server by default, except for the Solaris bundled installation, which does not include Java DB. Therefore, unless you have the Solaris bundled installation, you do not need to integrate this JDBC driver with the Communications Server.

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:** Java DB
- **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:javadb://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 GlassFish JDBC Driver for DB2 Databases

The JAR files for this driver are `smbase.jar`, `smbdb2.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 GlassFish JDBC Driver for Oracle 8.1.7 and 9.x 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 GlassFish 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 GlassFish 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 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/sqlllib/lib:${Java EE.home}/lib
DB2DIR=/opt/IBM/db2/V8.1
DB2INSTANCE=db2user
INSTHOME=/usr/db2user
WSPATH=/usr/db2user/sqlllib
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.

JConnect Type 4 Driver for Sybase ASE 12.5 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.SybXADDataSource
```

- **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`.

MySQL Type 4 Driver

The JAR file for the MySQL driver is `mysql-connector-java-version-bin-g.jar`, for example, `mysql-connector-java-5.0.5-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`.

```
jdbc:mysql://host:port/database
```

Inet Oraxo JDBC Driver for Oracle 8.1.7 and 9.x 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.
- **streamstoBlob** - 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.

Oracle Thin Type 4 Driver for Oracle 8.1.7 and 9.x 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 139](#).

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).

OCI Oracle Type 2 Driver for Oracle 8.1.7 and 9.x 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.OracleXDataSource
```

- **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.IfxXADataSource
```

- **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.

CloudScape 5.1 Type 4 Driver

The JAR files for the CloudScape driver are `db2j.jar`, `db2jtools.jar`, `db2jview.jar`, `jh.jar`, `db2jcc.jar`, and `db2jnet.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:** `Cloudscape`
- **DataSource Classname:** `com.ibm.db2.jcc.DB2DataSource`
- **Properties:**
 - **user** - Set as appropriate.
 - **password** - Set as appropriate.
 - **databaseName** - Set as appropriate.

Configuring Java Message Service Resources

The Communications Server implements the Java Message Service (JMS) API by integrating the Message Queue software into the Communications Server. For basic JMS API administration tasks, use the Communications Server Admin Console. For advanced tasks, including administering a Message Queue cluster, use the tools provided in the *MQ-as-install/imq/bin* directory. For details about administering Message Queue, see the *Message Queue Administration Guide*.

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

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 Communications Server, 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 *Java EE 5 Tutorial* provide details on these two types of communication and other aspects of JMS (see <http://java.sun.com/javaee/5/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 Communications 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 Communications Server deployment descriptor, link the physical destination name with the destination resource name.

The Relationship Between JMS Resources and Connector Resources

The Communications Server implements JMS by using a system resource adapter named `jmsra`. When a user creates JMS resources, the Communications 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 Communications Server creates a connector connection pool and connector resource. For each JMS destination a user creates, the Communications Server creates an admin object resource. When the user deletes the JMS resources, the Communications 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 [Chapter 7, “Connector Resources,”](#) for details.

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 Communications 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.

To manage JMS connection factories using the command-line utility, use `create-jms-resource`, `list-jms-resources`, or `delete-jms-resource` command.

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.

To manage JMS destinations using the command-line utility, use `create-jms-resource`, or `delete-jms-resource` command.

Tip – To specify the `addressList` property (in the format `host:mqport,host2:mqport,host3:mqport`) for `asadmin create-jms-resource` command, escape the `:` by using `\\`. For example, `host1\\:mqport,host2\\:mqport,host3\\:mqport`.

For more information on using escape characters, see the `asadmin(8)` man page.

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 > Java Message Service > 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.

To manage JMS physical destinations using the command-line utility, use `create-jmsdest`, `flush-jmsdest`, or `delete-jmsdest` command.

Configuring JMS Provider Properties

Use the JMS Service page in the Admin Console to configure properties to be used by all JMS connections. In the Admin Console, select Configurations > 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 Communications 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.

To manage JMS providers using the command-line utility, use the `set` or `jms -ping` commands.

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.

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 application server. The adapter is a .rar archive that can be deployed and configured using a Communications Server administration tools.

Configuring the Generic Resource Adapter for JMS

Communications 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 Communications 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 javabean classes as the mode of integration.

You can use the Admin Console or the command-line to configure the resource adapter. This is not different from configuring any other resource adapter.

Configuring the Generic Resource Adapter

Prior to deploying the resource adapter, JMS client libraries should be made available to the Communications 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 JVM(s).

1. Deploy the generic resource adapter the same way you would deploy a connector module.
2. Create a connector connection pool.
3. Create a connector resource.
4. Create an administered object resource.
5. Make the following changes to the security policy in the Communications Server:
 - Modify `sjsas_home/domains/domain1/config/server.policy` to add `java.util.logging.LoggingPermission "control"`
 - 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.

<i>Property Name</i>	<i>Valid Values</i>	<i>Default Value</i>	<i>Description</i>
ProviderIntegrationMode	javabean/jndi	javabean	Decides the mode of integration between the resource adapter and the JMS client.
ConnectionFactoryClassName	Name of the class available in the application server classpath, for example: <code>com.sun.messaging.ConnectionFactory</code>	None	Class name of <code>javax.jms.ConnectionFactory</code> implementation of the JMS client. Used if <code>ProviderIntegrationMode</code> is <code>javabean</code> .
QueueConnectionFactoryClassName	Name of the class available in the application server classpath, for example: <code>com.sun.messaging.QueueConnectionFactory</code>	None	Class name of <code>javax.jms.QueueConnectionFactory</code> implementation of the JMS client. Used if <code>ProviderIntegrationMode</code> is <code>javabean</code> .

<i>Property Name</i>	<i>Valid Values</i>	<i>Default Value</i>	<i>Description</i>
TopicConnectionFactory ClassName	Name of the class available in the application server 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 javabean.
XAConnectionFactory ClassName	Name of the class available in application server 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 javabean.
XAQueueConnection FactoryClassName	Name of the class available in application server 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 javabean.
XATopicConnection FactoryClassName	Name of the class available in application server classpath , for example: com.sun.messaging. XATopicConnectionFactory	None	Class name of javax.jms.XATopicConnectionFactory implementation of the JMS client. Used if ProviderIntegrationMode is javabean.
TopicClassName	Name of the class available in application server classpath , for example: com.sun.messaging.Topic	None	Class Name of javax.jms.Topic implementation of the JMS client. Used if ProviderIntegrationMode is javabean.
QueueClassName	Name of the class available in application server classpath , for example: com.sun.messaging.Queue	None	Class Name of javax.jms.Queue implementation of the JMS client. Used if ProviderIntegrationMode is specified as a javabean.
SupportsXA	True/false	FALSE	Specifies whether the JMS client supports XA or not.
ConnectionFactory Properties	Name value pairs separated by comma	None	Specifies the javabean property names and values of the ConnectionFactory of the JMS client. Required only if ProviderIntegrationMode is javabean.

<i>Property Name</i>	<i>Valid Values</i>	<i>Default Value</i>	<i>Description</i>
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> .
CommonSetter MethodName	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>javabean</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.
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	Provider Managed	<p>The <code>isSameRM</code> method on an <code>XAResource</code> is used by the Transaction Manager to determine if the Resource Manager instance represented by two <code>XAResources</code> are the same. When <code>RMPolicy</code> is set to <code>ProviderManaged</code> (the default value), the JMS provider is responsible for determining the <code>RMPolicy</code> and the <code>XAResource</code> wrappers in the Generic Resource Adapter merely delegate the <code>isSameRM</code> call to the message queue provider's <code>XA</code> resource implementations. This should ideally work for most message queue products.</p> <p>Some <code>XAResource</code> 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 <code>RMPolicy</code> is set to <code>OnePerPhysicalConnection</code>, the <code>XAResource</code> wrapper implementation's <code>isSameRM</code> in Generic Resource Adapter would check if both the <code>XAResources</code> 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.

<i>Property Name</i>	<i>Valid Value</i>	<i>Default Value</i>	<i>Description</i>
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 ProviderIntegrationMode 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.
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.
SendBadMessages ToDMD	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.
DeadMessage Destination JndiName	a valid 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> .
DeadMessage Destination ClassName	class name of destination object.	None	Used if <code>ProviderIntegrationMode</code> is <code>javabean</code> .
DeadMessage Destination Properties	Name Value Pairs separated by comma	None	Specifies the <code>javabean</code> property names and values of the destination of the JMS client. This is required only if <code>ProviderIntegrationMode</code> is <code>javabean</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 Communications 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 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.

The JavaMail API is implemented as a Java platform optional package and is also available as part of the J2EE platform.

The Communications Server includes the JavaMail API along with JavaMail service providers that allow an application component to send email notifications over the Internet and to read email from IMAP and POP3 mail servers.

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

This section contains the following topic:

Creating a JavaMail Session

To configure JavaMail for use in Communications Server, create a Mail Session in the Communications 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.
- **Additional Properties:** Create properties required by applications, such as a protocol-specific host or username property. Ensure that every property name you define start with a `mail-` prefix. If the name of the property does not start with `mail-`, the property is ignored.

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 Communications Server. To create a new resource, a new name-object binding is entered into the JNDI.

This section covers the following topics:

- “Java EE Naming Services” on page 79
- “Naming References and Binding Information” on page 80
- “Using Custom Resources” on page 81
- “Using External JNDI Repositories and Resources” on page 81

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. When it starts up, the Communications 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 decouples 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 J2EE applications from preexisting components.

The following table lists JNDI lookups and their associated references for the J2EE resources used by the Communications Server.

TABLE 6-1 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. In this section, we will discuss how to configure JNDI connection factory resources, for J2EE resources, and how to access these resources.

Within the Communications Server, you can create, delete, and list resources, as well as `list-jndi-entries`.

Using External JNDI Repositories and Resources

Often applications running on the Communications 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>
```

Connector Resources

This chapter explains how to configure connectors, which are used to access enterprise information systems (EISs). This chapter contains the following sections:

- “An Overview of Connectors” on page 83
- “Managing Connector Connection Pools” on page 84
- “Managing Connector Resources” on page 90
- “Managing Administered Object Resources” on page 92

An Overview of Connectors

Also called a resource adapter, a connector module is a Java EE 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. Like other Java EE modules, to install a connector module you deploy it.

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. (For more information on JNDI, see the section JNDI Names and Resources.) The JNDI name of a connector resource for an EIS is usually in the `java:comp/env/eis-specific` subcontext.

The Communications Server implements JMS by using a connector module (resource adapter). See the section, The Relationship Between JMS Resources and Connector Resources.

Managing Connector Connection Pools

- [“To Create a Connector Connection Pool” on page 84](#)
- [“To Edit a Connector Connection Pool” on page 85](#)
- [“To Edit Connector Connection Pool Advanced Attributes” on page 87](#)
- [“To Edit Connection Pool Properties” on page 89](#)

▼ To Create a Connector Connection Pool

Before You Begin Before creating the pool, deploy the connector module (resource adapter) associated with the pool. The values that are specified for the new pool depend on the connector module that is deployed.

- 1 In the tree component, expand the Resource node and then the Connectors node. Select the Connector Connection Pools node. On the Connector Connection Pools page, click New.**
- 2 On the first Create Connector Connection Pool page, specify the following settings:**
 - a. In the Name field, enter a logical name for the pool.**

Specify this name when creating a connector resource.
 - b. Select an entry from the Resource Adapter combo box.**

The combo box displays a list of deployed resource adapters (connector modules).
- 3 Select a value from the Connection Definition combo box.**

The choices in the combo box depend on the resource adapter you selected. The Connection Definition attribute identifies a resource adapter's connection-definition element in the `ra.xml` file.
- 4 Click Next.**
- 5 On the next Create Connector Connection Pool page, perform these tasks:**
 - a. In the General Settings section verify that the values are correct.**
 - b. For the fields in the Pool Settings section, the default values can be retained.**

These settings can be changed at a later time. See [“To Edit a Connector Connection Pool” on page 85](#).

c. In the Additional Properties table, add any required properties.

In the previous Create Connector Connection Pool page, you selected a class in the Connection Definition combo box. If this class is in the server's classpath, then the Additional Properties table displays default properties.

6 Click Finish.

More Information Equivalent asadmin command

```
create-connector-connection-pool
```

▼ To Edit a Connector Connection Pool

- 1 In the tree component, expand the Resources node and then the Connectors node. Expand the Connector Connection Pools node and select the connector connection pool you want to edit.**
- 2 On the Edit Connector Connection Pool page, you can change the settings of this connection pool. (Optional) Under General Settings, modify the description for the connection pool.**
- 3 Under Pool Settings, you can specify the parameter values as explained in the following table.**

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 scales up and scales down towards the maximum and minimum pool size respectively, it is resized in batches. This value determines the number of connections in the batch. Making this value too large will delay connection creation and recycling; making it too small will make it less efficient.
Idle Timeout	The maximum time in seconds that a connection can remain idle in the pool. After this time expires, the connection is removed from the pool.
Max Wait Time	The amount of time the application requesting a connection will wait before getting a connection timeout.

<p>On Any Failure</p>	<p>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.</p>
<p>Transaction Support</p>	<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. The transaction support options include the following. 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 XAResource or LocalTransaction 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. Local transaction support means that the resource adapter supports local transactions by implementing the LocalTransaction interface. Local transactions are managed internal to a resource manager and involve no external transaction managers. XA transaction support means that the resource adapter supports resource manager local and JTA transactions by implementing the LocalTransaction and XAResource 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>
<p>Connection Validation</p>	<p>Select the Required checkbox to enable connection validation.</p>

- 4 Click Save.**
- 5 Click Load Defaults if you want to restore the default values of all the settings.**

More Information Using the `asadmin` commands to change connection pool properties.

You can use the `asadmin get` and `set` commands to view and change the values of the connection pool properties.

To list all the connector connection pools in the server:

```
asadmin list domain.resources.connector-connection-pool.*
```

To view the properties of the connector connection pool, you can use the following command:

```
asadmin get domain.resources.connector-connection-pool.connectionpoolname.*
```

To set a property of the connector connection pool, you can use the following command:

```
asadmin set
domain.resources.connector-connection-pool.connectionpoolname.pool-resize-quantity=3
```

▼ To Edit Connector Connection Pool Advanced Attributes

1 Use the Advanced tab to edit the advanced attributes for a connector connection pool.

Parameter	Description
Validate Atmost Once	Amount of time, in seconds, after which a connection is validated at most once. This will help reduce the number of validation requests by a connection. The default value 0 implies that connection validation is not enabled.
Leak Timeout	Amount of time, in seconds, to trace connection leaks in a connection pool. The default value 0 means that connection leak tracing is disabled. If connection leak tracing is enabled, you can get statistics on the number of connection leaks in the Monitoring Resources tab. To view this tab, go to Application Server > Monitoring > Resources.
Leak Reclaim	If this option is enabled, leaked connections will be restored to the pool after leak connection tracing is complete.
Creation Retry Attempts	Number of attempts that will be made if there is a failure in creating a new connection. The default value of 0 implies that no attempts will be made to create the connection again.
Retry Interval	Specify the interval, in seconds, between two attempts to create a connection. The default value is 10 seconds. This attribute is used only if the value of Creation Retry Attempts is greater than 0.

Lazy Association	Connections are lazily associated when an operation is performed on them. Also, they are disassociated when the transaction is completed and a component method ends, which helps reuse of the physical connections. Default value is false.
Lazy Connection Enlistment	Enable this option to enlist a resource to the transaction only when it is actually used in a method.
Associate with Thread	Enable this option to associate a connection with the thread such that when the same thread is in need of a connection, it can reuse the connection already associated with it, thereby not incurring the overhead of getting a connection from the pool. Default value is false.
Match Connections	Use this option to switch on/off connection matching for the pool. It can be set to false if the administrator knows that the connections in the pool will always be homogeneous and hence a connection picked from the pool need not be matched by the resource adapter. Default value is false.
Max Connection Usage	Specify the number of times a connection should be reused by the pool. Once a connection is reused for the specified number of times, it will be closed. This is useful, for instance, to avoid statement-leaks. The default value of 0 implies that this attribute is not enabled.

- 2 Click **Save** for the changes to take effect.
- 3 Click **Load Defaults** if you want to restore the default values of all the settings.

More Information Using the `asadmin` commands to change connection pool properties.

You can use the `asadmin get` and `set` commands to view and change the values of the connection pool properties.

To list all the connector connection pools in the server:

```
asadmin list domain.resources.connector-connection-pool.*
```

To view the properties of the connector connection pool, you can use the following command:

```
asadmin get domain.resources.connector-connection-pool.connectionpoolname.*
```

To set a property of the connector connection pool, you can use the following command:

```
asadmin set
domain.resources.connector-connection-pool.connectionpoolname.validate-atmost-once-period-in-seconds=
```


▼ To Edit Connection Pool Properties

- 1 Use the **Additional Properties** tab to modify the properties of an existing pool.

The properties specified depend on the resource adapter used by this pool. The name-value pairs specified by the deployer using this table can be used to override the default values for the properties defined by the resource-adapter vendor.

- 2 Click **Save** for the changes to take effect.

Managing Security Maps

You can create, edit, and delete security maps associated with connector connection pools.

▼ To create security maps for connector connection pools

- 1 In the **Edit Connector Connection Pool** page, click the **Security Maps** tabbed pane.
- 2 Click **New** to create a new security map.
- 3 In the **New Security Map** page, provide a name for the security map, a user name, and password.
- 4 Click **OK**.

▼ To Edit Security Maps for Connector Connection Pools

- 1 In the **Edit Connector Connection Pool** page, click the **Security Maps** tabbed pane.
- 2 From the list of security maps, click the name of the security map that you want to edit.
- 3 Modify the values as desired, and click **Save**.

▼ To Delete a Connector Connection Pool

- 1 In the tree component, expand the **Resources** node and then the **Connectors** node.
- 2 Select the **Connector Connection Pools** node.
- 3 On the **Connector Connection Pools** page, select the checkbox for the pool to be deleted.
- 4 Click **Delete**.

More Information Equivalent asadmin command

`delete-connector-connection-pool`

▼ **To Set Up EIS Access**

- 1 Deploy (install) a connector.
- 2 Create a connection pool for the connector.
- 3 Create a connector resource that is associated with the connection pool.

Managing Connector Resources

- [“To Create a Connector Resource” on page 90](#)
- [“To Edit a Connector Resource” on page 91](#)
- [“To Delete a Connector Resource” on page 91](#)
- [“To Configure the Connector Service” on page 92](#)

▼ **To Create a Connector Resource**

A connector resource (data source) provides applications with a connection to an EIS.

Before You Begin Before creating a connector resource, first create a connector connection pool.

- 1 In the tree component, expand the Resources node and then the Connectors node.
- 2 Expand the Connector Resources node.
- 3 On the Connector Resources page, click New.
- 4 On the Create Connector Resources page, specify the resource’s settings:
 - a. In the JNDI Name field, type a unique name, for example: `eis/myERP`.
 - b. From the Pool Name combo box, choose the connection pool to which the new connector resource belongs.
 - c. To change the resource to be unavailable, deselect the Status checkbox.
By default, the resource is available (enabled) when it is created.

d. Do one of the following:

- If you are using the cluster profile, you will see the Targets section of the page. In the Targets section of the page, select the domain, cluster, or server instances where the connector resource will reside, from the Available field and click Add. If you do not want to deploy the connector resource to one of the domains, clusters, or server instances listed in the Selected field, select it from the field and click Remove.
- If you are running a domain with developer profile, skip to step 5.

5 Click OK.

More Information Equivalent `asadmin` command

```
create-connector-resource
```

▼ To Edit a Connector Resource

1 In the tree component, expand the Resources node and then the Connectors node.

2 Expand the Connector Resources node.

3 Select the node for the connector resource that you want to edit. On the Edit Connector Resources page, you can change the settings of this connection.

4 Do one of the following:

- If you are using the cluster profile or enterprise profile, you will see the Targets tabbed pane. On the Targets tabbed pane, you can enable or disable the connector resource on individual targets. You can change the targets on which the connector resource is deployed by clicking Manage Targets.
- If you are using developer profile, skip to next step.

5 Click Save to apply the edits.

▼ To Delete a Connector Resource

1 In the tree component, expand the Resources node and then the Connectors node.

2 Select the Connector Resources node.

- 3 On the **Connector Resources** page, select the checkbox for the resource to be deleted.
- 4 Click **Delete**.

More Information Equivalent `asadmin` command

```
delete-connector-resource
```

▼ To Configure the Connector Service

Use the Connector Service screen to configure the connector container for all resource adapters deployed to this cluster or server instance.

- 1 **Select Configurations from the tree.**
- 2 **If you are using the cluster profile, select the instance to configure:**
 - **To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.**
 - **To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.**
- 3 **Select the Connector Service node.**
- 4 **Specify the shutdown timeout in seconds in the Shutdown Timeout field.**

Enter an integer representing the number of seconds that the application server waits to allow the `ResourceAdapter.stop` method of the connector module's instance to complete. Resource adapters that take longer than the specified shutdown timeout are ignored by the application server and the shutdown procedure continues. The default shutdown timeout is 30 seconds. Click **Load Defaults** to select the default shutdown timeout for the resource adapters deployed to this cluster or server instance.

Managing Administered Object Resources

Packaged within a resource adapter (connector module), an administered object provides specialized functionality for an application. For example, an administered object might provide access to a parser that is specific to the resource adapter and its associated EIS. The object can be administered; that is, it can be configured by an administrator. To configure the object, add name-value property pairs in the **Create or Edit Admin Object Resource** pages. When creating an administered object resource, associate the administered object to a JNDI name.

To create, edit, and delete Connector Connection Pools, click Resources —> Administered Object Resources in the Admin Console. Consult the Admin Console Online Help for detailed instructions on managing connector connection pools.

- [“To Create an Administered Object Resource” on page 93](#)
- [“To Edit an Administered Object Resource” on page 94](#)
- [“To Delete an Administered Object Resource” on page 94](#)

▼ To Create an Administered Object Resource

Packaged within a resource adapter (connector module), an administered object provides specialized functionality for an application. For example, an administered object might provide access to a parser that is specific to the resource adapter and its associated EIS. The object can be administered; that is, it can be configured by an administrator. To configure the object, add name-value property pairs in the Create or Edit Admin Object Resource pages. When creating an administered object resource, associate the administered object to a JNDI name.

The Communications Server implements JMS by using resource adapter. For each JMS destination created, the Communications Server automatically creates an administered object resource.

- 1 In the tree component, expand the Resources node and then the Connectors node.**
- 2 Expand the Admin Object Resources node.**
- 3 On the Admin Object Resources page, click New.**
- 4 On the Admin Object Resources page, specify the following settings:**
 - a. In the JNDI Name field, type a unique name that identifies the resource.**
 - b. In the Resource Type field, enter the Java type for the resource.**
 - c. From the Resource Adapter combo box, select the resource adapter that contains the administered object.**
 - d. Select or deselect the Status checkbox to enable or disable the resource.**
 - e. To configure the administered object with name-value property pairs, click Add Property.**

- f. If you are using cluster profile, in the Targets section of the page, select the domain, cluster, or server instances where the administered object will reside, from the Available field and click Add.

To undeploy the administered object to one of the domains, clusters, or server instances listed in the Selected field, select it from the field and click Remove.

- g. Click Finish.

More Information Equivalent `asadmin` command

`create-admin-object`

▼ To Edit an Administered Object Resource

- 1 In the tree component, expand the Resource node and then the Connectors node.
- 2 Expand the Administered Object Resources node.
- 3 Select the node for the administered object resource to be edited.
- 4 On the Edit Administered Object Resources page, modify values specified in Creating an Administered Object Resource.
- 5 If you are using cluster profile, on the Targets tabbed pane, edit the targets on which the administered object is deployed by clicking Manage Targets.
- 6 Click Save to apply the edits.

▼ To Delete an Administered Object Resource

- 1 In the tree component, expand the Resources node and then the Connectors node.
- 2 Select the Administered Object Resources node.
- 3 On the Administered Object Resources page, select the checkbox for the resource to be deleted.
- 4 Click Delete.

More Information Equivalent `asadmin` command

`delete-admin-object`

SIP, Web, and EJB Containers

Containers provide runtime support for application components. Application components use the protocols and methods of the container to access other application components and services provided by the server. The Communications 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 [“Communications Server Overview” on page 25](#).

This chapter describes the following containers:

- [“The SIP Servlet Container” on page 95](#)
- [“The Web Container” on page 98](#)
- [“The EJB Container” on page 98](#)

The SIP Servlet Container

Communications Server provides the SIP Servlet container that hosts SIP-compliant applications. Features of this container include the following:

- Provides a network end point to listen to SIP requests.
- Provides an environment to host and manage the lifecycle of SIP Servlets.
- Decides what applications to host in which order.
- Supports the Secure SIP protocol, SIPS over the transport layer protocol TLS.
- Uses Grizzly NIO framework for server side socket listeners.
- Implements Digest Authentication for security.
- Supports Call Flow. Call Flow is the feature that allows application developers and Application Server administrators to monitor the behavior of the deployed applications.

Editing the Properties of the SIP Container

To view SIP container settings on the default domain, use the following `asadmin` command:

```
get server.sip-container.*
```

Typical output is as follows:

```
server.sip-container.external-address =  
server.sip-container.external-sip-port = 5060  
server.sip-container.external-sips-port = 5061p
```

By default, the external address is empty. The SIP Container has a heuristic that tries to find a suitable address by examining all network interfaces on the machine. In a cluster setup, it is mandatory to change the external address. The external SIP port and SIPS port listen at 5060 and 5061, by default. Changing the SIP and SIPS port settings of the SIP listeners does not change the external SIP port or SIPS port of the SIP container. To change the port settings of the SIP container, use the following `asadmin` command:

```
set server.sip-container.external-sip-port=5065
```

The sub-elements of the SIP container are `session-manager` and `session-properties`. `store-properties`, `manager-properties` are sub-elements of `session-manager`.

For a complete list of SIP container properties, see [TBDlink](#).

To change the properties of the SIP container using the Admin Console, select the Configuration node, select the configuration name, and the SIP Container node. Use the General, Session Timeout, SIP Container Manager, or SIP Container Store tabs to view and change SIP container properties.

You can use the following CLI commands to list and view the sub-elements and attributes of the SIP container.

To view the sub-elements of the SIP container, use the following command: `list`

```
server.sip-container.*
```

To view the attributes of the SIP container, use the following command: `get`

```
server.sip-container.*
```

Editing SIP Container General Attributes

To view the attributes of the SIP container, use the following command: `get`

```
server.sip-container.*
```

- External Address
- SIP Port
- Secure SIP Port

To get the attributes of the SIP container, use the get and set commands:

```
get server.sip-container.external-sip-port
set server.sip-container.external-sip-port=5060
```

Editing SIP Container Session Properties

To view the attributes of the SIP container session properties, use the following command: `get server.sip-container.session-config.session-properties.*`

To set the attributes of the SIP container session properties, use the get and set commands:

```
get server.sip-container.session-config.session-properties.*
set
server.sip-container.session-config.session-properties.timeout-in-seconds=0
```

Editing SIP Container Session Manager Properties

To view the sub-elements of the SIP container session manager properties, use the following command: `list server.sip-container.session-config.session-manager.*`

The two sub-elements are `store-properties`, `manager-properties`.

To view the attributes of `store-properties`, use the following command: `get server.sip-container.session-config.session-manager.store-properties.*`

To view the attributes of `manager-properties`, use the following command: `get server.sip-container.session-config.session-manager.manager-properties.*`

To set the attributes of the SIP container session manager properties, use the get and set commands:

```
get
server.sip-container.session-manager.manager-properties.reap-interval-in-seconds
set
server.sip-container.session-manager.manager-properties.reap-interval-in-seconds=2
```

For a complete list of SIP container properties, see the [TBDlink](#),

The Web Container

The Web Container is a J2EE container that hosts web applications. The web container extends the web server functionality by providing developers the environment to run servlets and 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.

Configuring Security

Security is about protecting data: how to prevent unauthorized access or damage to it in storage or transit. The Communications Server has a dynamic, extensible security architecture based on the Java EE standard. Built-in security features include cryptography, authentication and authorization, and public key infrastructure. The Communications 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 99
- “Tools for Managing Security” on page 100
- “Managing Security of Passwords” on page 101
- “About Authentication and Authorization” on page 104
- “Understanding Users, Groups, Roles, and Realms” on page 106
- “Trust Configurations and Entities” on page 110
- “Introduction to Certificates and SSL” on page 111
- “About Firewalls” on page 114
- “About Certificate Files” on page 114
- “Using Java Secure Socket Extension (JSSE) Tools” on page 115

Understanding Application and System Security

Broadly, there are two kinds of application security:

- In *programmatically 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 Java EE containers.
- In *declarative security*, the container (the Communications 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 Communications 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 Communications 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 26. For an overview of the security tasks consult the Admin Console online help.
- `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 from a script, to automate repetitive tasks. For a general introduction to `asadmin`, see “[Tools for Administration](#)” on page 26.

The Java Platform, Standard Edition (Java SE) 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 *JDK Tools and Utilities* at <http://java.sun.com/j2se/1.5.0/docs/tooldocs/#security>.

In the Enterprise Profile, two other tools that implement Network Security Services (NSS) are available for managing security. For more information on NSS, go to <http://www.mozilla.org/projects/security/pki/nss/>. The tools for managing security include the following:

- `certutil`, a command-line utility for managing certificates and key databases.
- `pk12util`, a command-line utility used to import and export keys and certificates between the certificate/key databases and files in PKCS12 format.

Managing Security of Passwords

In Communications Server, the file `domain.xml`, which contains the specifications for a particular domain, initially contains the password of the 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. Instructions for managing the security passwords is included in the following topics:

- “[Encrypting a Password in the domain.xml File](#)” on page 101
- “[Protecting Files with Encoded Passwords](#)” on page 102
- “[Changing the Master Password](#)” on page 102
- “[Working with the Master Password and Keystores](#)” on page 103
- “[Changing the Admin Password](#)” on page 103

Encrypting a Password in the `domain.xml` File

To encrypt a password in the `domain.xml` file. Follow these steps:

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`. This is accomplished using the `asadmin set` command. An example of using the `set` command for this purpose is as follows:

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

Note – Enclose the alias password in single quotes as shown in the example.

3. Restart the Communications 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 permissions 600.
- Any password file created to pass as an argument using the `--passwordfile` argument to `asadmin` should be protected with file system permissions 600.

Changing 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 central 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 master password by removing this file. When the master password is changed, it is re-saved in the master-password keystore, which is a Java JCEKS type keystore.

To change the master password, follow these steps:

1. Stop the Communications 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 master password>
Please enter the new master password>
Please enter the the new master password again>
```

2. Restart the Communications Server.



Caution – At this point in time, server instances that are running must not be started and running server instances must not be restarted until the SMP on their corresponding node agent has been changed. If a server instance is restarted before changing its SMP, it will fail to come up.

3. Stop each node agent and its related servers one at a time. Run the `asadmin change-master-password` command again, and then restart the node agent and its related servers.
4. Continue with the next node agent until all node agents have been addressed. In this way, a rolling change is accomplished.

Working with the Master Password and Keystores

The master password is the password for the secure keystore. When a new application server domain is created, a new self-signed certificate is generated and stored in the relevant keystore, which is locked using the master password. If the master password is not the default, the `start-domain` command prompts you for the master password. Once the correct master password is entered, the domain starts.

When a node agent associated with the domain is created, the node agent synchronizes the data with domain. While doing so, the keystore is also synchronized. Any server instance controlled by this node agent needs to open the keystore. Since the store is essentially identical to the store that was created by the domain creation process, it can only be opened by an identical master password. But the master password itself is never synchronized, meaning it is not transmitted to the node agent during the synchronization, but needs to be available with the node agent locally. This is why creation and/or starting of a node agent prompts you for the master password and you need to enter the same password that you entered while creating/starting the domain. If the master password is changed for a domain, you will have to perform the same step to change it at every node agent that is associated with this domain.

Changing the Admin Password

Encrypting the admin password is discussed in “[Managing Security of Passwords](#)” on page 101. Encrypting the admin password is strongly encouraged. If you want to change the admin password before encrypting it, use the `change-admin-password` command.

Consult the Admin Console online help for instructions on changing the admin password using the Admin Console.

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 104
- “Authorizing Users” on page 105
- “Specifying JACC Providers” on page 105
- “Auditing Authentication and Authorization Decisions” on page 105
- “Configuring Message Security” on page 106

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, a digital certificate, or something else.

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 Communications 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 Communications Server supports four types of authentication. An application specifies the type of authentication it uses within its deployment descriptors.

TABLE 9-1 Communications 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	HTTP (SSL optional)	Application provides its own custom login and error pages.	None, unless using SSL.
CLIENT-CERT	HTTPS (HTTP over SSL)	Server authenticates the client using a public key certificate.	SSL

TABLE 9-1 Communications Server Authentication Methods (Continued)

DIGEST	HTTP and SIP	Server authenticates the client based on an encrypted response.	SSL and TLS
--------	--------------	---	-------------

Verifying Single Sign-On

Single sign-on enables multiple applications in one virtual server instance to share the 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, CLIENT-CERT) share single sign-on.

Single sign-on is enabled by default for virtual servers defined for the Communications Server.

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 106.

Specifying JACC Providers

JACC (Java Authorization Contract for Containers) is part of the Java EE 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 Communications 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 Communications Server can provide an audit trail of all authentication and authorization decisions through *audit modules*. The Communications Server provides a default audit module, as well as the ability to customize the audit modules.

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

Message level security can be configured for the entire Communications Server or for specific applications or methods. Configuring message security at the Communications Server level is discussed in [Chapter 10, “Configuring Message Security.”](#) Configuring message security at the application level is discussed in the *Developer's Guide*.

Understanding Users, Groups, Roles, and Realms

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

- **“Users” on page 107:** An individual identity *defined in the Communications 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 107:** A set of users *defined in the Communications Server*, classified by common traits.
- **“Roles” on page 108:** 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 108:** 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 Communications 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.

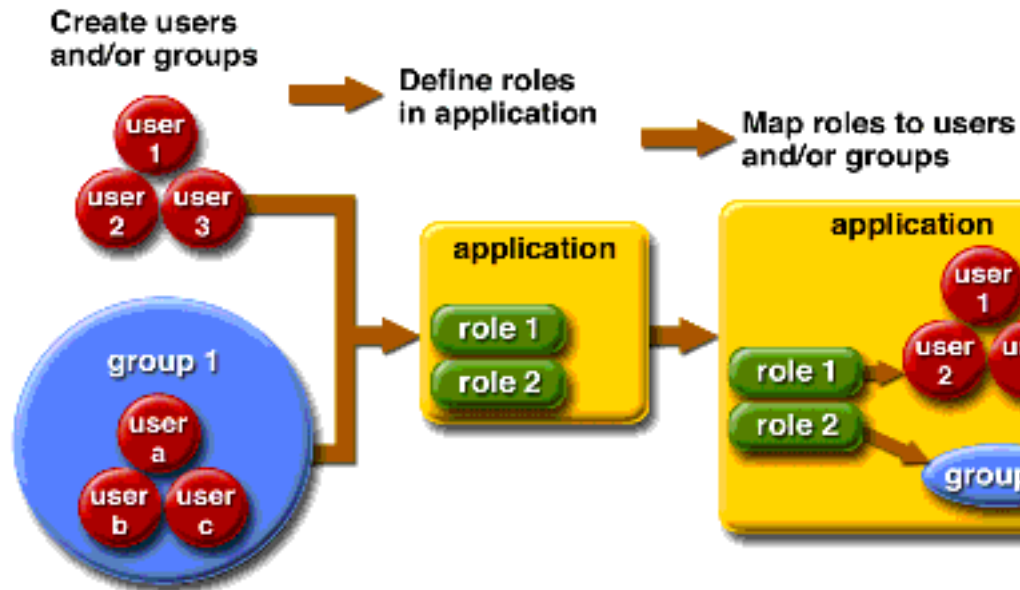


FIGURE 9-1 Role Mapping

Users

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

Groups

A *Java EE 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 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.

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

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 111](#).

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.

In the `ldap` realm the server gets user credentials from a Lightweight Directory Access Protocol (LDAP) server such as the Directory Server. LDAP is a protocol for enabling anyone 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 JDBC realm, the server gets user credentials from a database. The Communications Server uses the database information and the enabled JDBC realm option in the configuration file. For digest authentication, a JDBC realm should be created with `jdbcDigestRealm` as the JAAS context.

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 the Admin Console online help.

▼ To Configure a JDBC Realm for a Web, EJB, or SIP Application

The Communications Server enables you to specify a user's credentials in the JDBC realm instead of in the connection pool. Using the JDBC realm instead of the connection pool prevents other applications from browsing the database tables for the user's credentials. A user's credentials are the user's name and password.

Note – By default, storage of passwords as clear text is not supported in the JDBC realm. Under normal circumstances, passwords should not be stored as clear text.

1 Create the database tables in which to store the users' credentials for the realm.

How to create the database tables depends on the database that you are using.

2 Add the users' credentials to the database tables that you created in [Step 1](#).

How to add users' credentials to the database tables depends on the database that you are using.

3 Create a JDBC realm.

Use the Admin Console GUI for this purpose. For instructions for creating a JDBC realm, see the online help for the Admin Console GUI.

4 Specify the realm that you created in [Step 3](#) as the realm for the application.

To specify the realm, modify the appropriate deployment descriptor for your application:

- **For an enterprise application in an Enterprise Archive (EAR) file, modify the `sun-application.xml` file.**
- **For a web application in a Web Application Archive (WAR) file, modify the `web.xml` file.**
- **For a SIP application, modify the `sun-sip.xml` file.**
See [TBDlink](#) for more details.

- **For an enterprise bean in an EJB JAR file, modify the `sun-ejb-jar.xml` file.**

For more information about how to specify a realm, see “[How to Set a Realm for an Application or Module](#)” in *Sun GlassFish Communications Server 1.5 Developer’s Guide*.

5 Assign a security role to users in the realm.

To assign a security role to a user, add a `security-role-mapping` element to the deployment descriptor that you modified in [Step 4](#).

The following example shows a `security-role-mapping` element that assigns the security role `Employee` to user `Calvin`.

```
<security-role-mapping>
  <role-name>Employee</role-name>
  <principal-name>Calvin</principal-name>
</security-role-mapping>
```

Trust Configurations and Entities

You can create configurations to set up a trust relationship among domains and/or hosts. In a configuration, you can specify the trust type, the host or domains that are trusted and if the trust relationship is created for sending or receiving messages.

Creating Trust Configurations and Entities

You can create an identity assertion trust by using the Admin Console or CLI.

In the Admin Console, expand the Configurations node and select a configuration. Expand the Security node, click Trust Configurations and then click New.

Alternatively, you can use the `asadmin create-trusted-entity` and `asadmin create-trust-config` command. See [TBDlink](#) for more details about this command.

You can associate trusted entities (trusted hosts/domains) with a trust configuration or you can select a Trust Handler to have custom implementations to determine trust.

Editing Trust Configurations and Entities

To edit the properties of the trust configuration, you can use the Admin Console or you can use `list`, `get`, and `set` commands as follows:

To list the trust configurations in a target, use the command: `list config-name.security-service.identity-assertion-trust.*`

To get the trust configuration attributes, use the command: `get config-name.security-service.identity-assertion-trust.trust-config-name.*`

For a complete list of trust configuration attributes, see TBDlink.

To list the trusted entity in a configuration, use the command: `list config-name.security-service.identity-assertion-trust.trust-config-name.*`

To edit the properties of the trusted entity, you can use the Admin Console or you can use `list`, `get`, and `set` commands as follows:

To get the trusted entity attributes, use the command: `get config-name.security-service.identity-assertion-trust.trust-config-name.trusted-entity.t`

To set the trusted entity attributes, use the command: `set config-name.security-service.identity-assertion-trust.trust-config-name.trusted-entity.t`

For a complete list of trusted entity attributes, see TBDlink.

Introduction to Certificates and SSL

The following topics are discussed in this section:

- “About Digital Certificates” on page 111
- “About Secure Sockets Layer” on page 113

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 113.

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 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* happens:

- 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 Communications Server supports the Secure Sockets Layer (SSL) 3.0 and the Transport Layer Security (TLS) 1.0 encryption protocols.

To use SSL, the Communications 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 [“Generating a Certificate Using the keytool Utility” on page 117](#) 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 Communications 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 Communications Server behind the firewall.

About Certificate Files

Installation of the Communications Server generates a digital certificate in JSSE (Java Secure Socket Extension) or NSS (Network Security Services) format suitable for internal testing. By default, the Communications Server stores its certificate information in a certificate database in the *domain-dir/config* directory:

- **Keystore file**, `key3.db`, contains the Communications Server's certificate, including its private key. The keystore file is protected with a password. Change the password using the `asadmin change-master-password` command.

Each keystore entry has a unique alias. After installation, the Communications Server keystore has a single entry with alias `s1as`.

- **Truststore file**, `cert8.db`, contains the Communications 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 Developer Profile, on the server side, the Communications Server uses the JSSE format, which uses `keytool` to manage certificates and key stores. In the Clusters and Enterprise Profile, on the server side, the Communications Server uses NSS, which uses `certutil` to manage the NSS database which stores private keys and certificates. In both profiles, the client side (applient or stand-alone), uses the JSSE format.

By default, the Communications 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.

Changing the Location of Certificate Files

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

Use the Admin Console to add or modify the value field for the new location of the certificate files.

```
-Dcom.sun.appserv.nss.db=${com.sun.aas.instanceRoot}/NSS-database-directory
```

where `NSS-database-directory` is the location of the NSS database.

Using Java Secure Socket Extension (JSSE) Tools

Use `keytool` to set up and work with JSSE (Java Secure Socket Extension) digital certificates. In the Developer Profile, the Communications Server uses the JSSE format on the server side to manage certificates and key stores. In all the profiles, the client side (applient 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}
-dname ${dn.name} -keypass ${key.pass} -keystore ${keystore.file}
-storepass ${keystore.pass}
```

Another example of creating a certificate is shown in “[Generating a Certificate Using the `keytool` Utility](#)” on page 117.

- 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 “[Signing a Digital Certificate Using the `keytool` Utility](#)” on page 118

- 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 119](#)

Generating 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 [“Changing the Location of Certificate Files” on page 115](#).

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. The default key password alias is “s1as.”

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 [“Signing a Digital Certificate Using the keytool Utility” on page 118](#).
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
```

6. 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.

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

Then `keytool` displays something like this:

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

8. Restart the Communications Server.

Signing 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 [“Changing the Location of Certificate Files” on page 115](#).

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 changeit
```

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 Communications 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
```


Configuring Message Security

Some of the material in this chapter assumes a basic understanding of security and web services concepts. This chapter describes the configuration of message layer security for web services in the Communications Server. This chapter contains the following topics:

- “Overview of Message Security” on page 121
- “Understanding Message Security in the Communications Server” on page 122
- “Securing a Web Service” on page 126
- “Securing the Sample Application” on page 127
- “Configuring the Communications Server for Message Security” on page 127
- “Message Security Setup” on page 131

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 5.0 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>.

Understanding Message Security in the Communications Server

The Communications 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 Communications 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 Communications 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 Communications Server, the [“System Administrator” on page 122](#) and [“Application Deployer” on page 123](#) roles are expected to take primary responsibility for configuring message security. In some situations, the [“Application Developer” on page 123](#) 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 122](#)
- [“Application Deployer” on page 123](#)
- [“Application Developer” on page 123](#)

System Administrator

The system administrator is responsible for:

- Configuring message security providers on the Communications Server.
- Managing user databases.
- Managing keystore and truststore files.
- Configuring a Java Cryptography Extension (JCE) provider if using encryption and running a version of the Java SDK prior to version 1.5.0.
- 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 Communications Server for Message Security” on page 127](#).

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.

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 Communications 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 Communications Server.

About Username Tokens

The Communications 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 Communications Server

About Digital Signatures

The Communications 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 Communications Server. For more information on this topic, read [“About Certificate Files” on page 114](#).

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 [“Configuring a JCE Provider” on page 129](#).

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 Communications Server for Message Security](#)” on page 127.

- Authentication Layer

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

- Authentication Provider

In this release of the Communications Server, the Communications 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 Communications 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.

Securing a Web Service

Web services deployed on the Communications 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 Communications 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 Communications Server is installed, SOAP layer message security providers are configured in the client and server-side containers of the Communications 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 Communications Server can be employed to bind the existing providers for use by the server-side containers of the Communications Server, to modify the message protection policies enforced by the providers, or to create new provider configurations with alternative message protection policies. Analogous administrative operations can be performed on the SOAP message layer security configuration of the application client container as defined in [“Enabling Message Security for Application Clients” on page 133](#).

By default, message layer security is disabled on the Communications Server. To configure message layer security for the Communications Server follow the steps outlined in [“Configuring the Communications Server for Message Security” on page 127](#). If you want to cause web services security to be used to protect all web services applications deployed on the Communications Server, follow the steps in [“Enabling Providers for Message Security” on page 131](#).

Once you have completed the above steps (which may include restarting the Communications Server), web services security will be applied to all web services applications deployed on the Communications 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 [Chapter 5, “Securing Applications,” in *Sun GlassFish Communications Server 1.5 Developer’s Guide*](#).

Securing the Sample Application

The Communications 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 Communications 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 Communications 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 127](#) application).

The `xms` sample application is installed in the directory:
`as-install/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 Communications Server for Message Security

- [“Actions of Request and Response Policy Configurations” on page 128](#)
- [“Configuring Other Security Facilities” on page 129](#)
- [“Configuring a JCE Provider” on page 129](#)

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 10-1 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 <code>wsse:Security</code> header that contains a <code>wsse:UsernameToken</code> (with password).
auth-source="content"	The content of the SOAP message Body is signed. The message contains a <code>wsse:Security</code> header that contains the message Body signature represented as a <code>ds:Signature</code> .
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 <code>xenc:EncryptedData</code> . The message contains a <code>wsse:Security</code> header that contains a <code>wsse:UsernameToken</code> (with password) and 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.
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.

TABLE 10-1 Message protection policy to WS-Security SOAP message security operation mapping
(Continued)

Message Protection Policy	Resulting WS-Security SOAP message protection operations
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.

Configuring Other Security Facilities

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

1. If using a version of the Java SDK prior to version 1.5.0, and using encryption technology, configure a JCE provider.
2. Configuring a JCE provider is discussed in [“Configuring a JCE Provider” on page 129](#).
3. 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.
4. Manage certificates and private keys, if necessary.

After You Finish

Once the facilities of the Communications Server are configured for use by message security providers, then the providers installed with the Communications Server may be enabled as described in [“Enabling Providers for Message Security” on page 131](#).

Configuring 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 Communications Server on version 1.5 of the Java SDK, the JCE provider is already configured properly. If you are running the Communications 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/javase_providers.html.

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

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

Message Security Setup

Most of the steps for setting up the Communications 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 Communications Server from running properly, therefore, where possible, steps for configuring the Communications 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 Communications Server and its client containers in the form of (pluggable) authentication modules. By default, message layer security is disabled on the Communications Server. The following sections provide the details for enabling, creating, editing, and deleting message security configurations and providers.

- [“Enabling Providers for Message Security” on page 131](#)
- [“Configuring the Message Security Provider” on page 132](#)
- [“Creating a Message Security Provider” on page 133](#)
- [“Enabling Message Security for Application Clients” on page 133](#)
- [“Setting the Request and Response Policy for the Application Client Configuration” on page 133](#)
- [“Further Information” on page 134](#)

In most cases, it will be necessary to restart the Communications 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 Communications Server at the time the operation was performed.

Enabling Providers for Message Security

To enable message security for web services endpoints deployed in the Communications 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 Communications Server. Information for enabling the providers used by clients is discussed in [“Enabling Message Security for Application Clients” on page 133](#).

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 Communications Server, you must ensure that any services invoked from endpoints deployed in the Communications Server are compatibly configured for message layer security.

Use the command-line utility:

- 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
```

Configuring the Message Security Provider

Typically, a provider would be re-configured to modify its message protection policies, although the provider type, implementation class, and provider-specific configuration properties may also be modified.

Use the command-line utility 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 using the Admin Console, select Configuration node > the instance to Configure > Security node > Message Security node > SOAP node > Providers tab.

For more detailed instructions on creating a message security provider, see the Admin Console online help.

Enabling 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 Communications Server is installed.

To enable message security for client applications, modify the Communications Server specific configuration for the application client container.

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 Communications Server specific configuration for the application client container as described in [“Enabling Message Security for Application Clients” on page 133](#). In the application client configuration file, add the request-policy and response-policy elements as shown to set the request policy.

The other code is provided for reference. The other code 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="as-install/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 128](#).

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/1.4.2/docs/guide/security/index.html>.
- The *Java EE 5.0 Tutorial* chapter titled *Security* can be viewed from <http://java.sun.com/javaee/5/docs/tutorial/doc/index.html>.
- The *Administration Guide* chapter titled .
- The *Developer's Guide* chapter titled *Securing Applications*.
- 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/>.

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 135](#)
- [“Diagnostic Service Framework” on page 135](#)

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`, and `sun-sip.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 Administration Console. Confidential data appears in the generated report as listed in the Confidential Properties table.

Transactions

By enclosing one or more steps in an indivisible unit of work, a transaction ensures data integrity and consistency. This chapter contains the following sections:

- “About Transactions” on page 137
- “Admin Console Tasks for Transactions” on page 139

About Transactions

- “What is a Transaction?” on page 137
- “Transactions in Java EE Technology” on page 138
- “Workarounds for Specific Databases” on page 139

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.

See Also:

- [“Transactions in Java EE Technology” on page 138](#)
- [“Configuring Transactions” on page 139](#)

Transactions in Java EE Technology

Transaction processing involves the following five participants:

- Transaction Manager
- Communications Server
- Resource Manager(s)
- Resource Adapter(s)
- User Application.

Each of these entities contribute to reliable transaction processing by implementing the different APIs and functionalities, 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 Communications 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 Transactional User Application developed to operate in an application server environment looks up transactional data sources and, optionally, the transaction manager, using JNDI. The application may use declarative transaction attribute settings for enterprise beans or explicit programmatic transaction demarcation.

See Also:

- [“What is a Transaction?” on page 137](#)
- [“Configuring Transactions” on page 139](#)

Workarounds for Specific Databases

The Communications 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 Communications Server workaround, set the `oracle-xa-recovery-workaround` property value to `false`. For details about how to set a property, see [“To configure how the Communications Server recovers from transactions” on page 139](#).

Note – These workarounds do not imply support for any particular JDBC driver.

Admin Console Tasks for Transactions

The Communications Server handles transactions based on the settings in the Admin Console.

Configuring Transactions

This section explains how to configure transaction settings:

- [“To configure how the Communications Server recovers from transactions” on page 139](#)
- [“To set a transaction timeout value” on page 140](#)
- [“To set the location of the transaction logs” on page 141](#)
- [“To set the keypoint interval” on page 142](#)

For additional information about transactions, see these sections:

- [“What is a Transaction?” on page 137](#)
- [“Transactions in Java EE Technology” on page 138](#)

▼ **To configure how the Communications Server recovers from 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. Communications 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.

- 1 In the tree component select the Configurations node.**
- 2 If you are running a non-clustered domain in developer profile, skip to the next step. If you are running the domain in cluster profile, select the instance to configure:**
 - **To configure a particular instance, select the instance's config node. For example, the default instance, server, select the server-config node.**
 - **To configure the default settings for all instances, select the default-config node.**
- 3 Select the Transaction Service node.**
- 4 To enable the recovery of incomplete transactions, check the Recover in the On Restart field.**
- 5 Set the amount of time, in seconds, the Communications Server tries to connect to the unreachable server in the Retry Timeout field. The default value is 10 minutes (600 seconds).**
- 6 Set the policy for unreachable servers in a transaction in the Heuristic Decision field.**

Unless there is a good reason to set this field to Commit, leave Heuristic Decision set to Rollback. Committing indeterminate transactions can compromise the data integrity of your application.
- 7 Set any needed properties.**

Click the Add Properties button, type values in the Name and Value fields, and check the box to the left of the Name to activate the property.
- 8 Click Save.**
- 9 Restart the Communications Server.**

▼ **To set a transaction timeout value**

By default, the server does not timeout 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 Communications Server rolls back the transaction.

- 1 In the tree component, select the Configurations node.**

- 2 If you are running a non-clustered domain in developer profile, skip to the next step. If you are running the domain in cluster profile, select the instance to configure:
 - To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
 - To configure the default settings for all instances, select the `default-config` node.
- 3 Select the Transaction Service node.
- 4 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.
- 5 Click Save.
- 6 Restart the Communications Server.

▼ To set the location of 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.

- 1 In the tree component, select the Configurations node.
- 2 If you are running a non-clustered domain in developer profile, skip to the next step. If you are running the domain in cluster profile, select the instance to configure:
 - To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
 - To configure the default settings for all instances, select the `default-config` node.
- 3 Select the Transaction Service node.
- 4 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.

The default value is `${com.sun.aas.instanceRoot}/logs`. The `${com.sun.aas.instanceRoot}` variable is the name of the instance, and is set when you start an Communications Server instance. To see the value of `${com.sun.aas.instanceRoot}`, click Actual Values.
- 5 Click Save.

6 Restart the Communications Server.**▼ To set 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.

- 1 In the tree component select the Configurations node.**
- 2 If you are running a non-clustered domain in developer profile, skip to the next step. If you are running the domain in cluster profile, select the instance to configure:**
 - **To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.**
 - **To configure the default settings for all instances, select the `default-config` node.**
- 3 Select the Transaction Service node.**
- 4 Enter the number of transactions between keypoint operations in the Keypoint Interval field.**
The default value is 65536.
- 5 Click Save.**
- 6 Restart the Communications Server.**

Configuring the HTTP Service

The HTTP service is the component of the Communications 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.

This chapter discusses the following topics:

- “Virtual Servers” on page 143
- “HTTP Listeners” on page 144

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 Communications Server is running.

Note – Do not confuse an Internet domain with the administrative domain of the Communications 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 Communications 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 Communications 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 Communications 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`.
- An HTTP listener named `admin-listener`, associated with the virtual server named `__asadmin`.

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 Communications Server. If the Communications Server uses the default port number values, `http-listener-1` uses port 8080, `http-listener-2` uses port 8181, and `admin-listener` uses port 48489.

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 accept new connections and put them onto a connection queue. Session threads then pick up connections from the queue and service the requests. The server posts more session threads if required at the end of the request.

A set of request processing threads retrieves incoming HTTP requests from the connection queue and processes the requests. These threads parse the HTTP headers, select the appropriate virtual server, and run through the request processing engine to service the request. When there are no more requests to process, but the connection can be kept persistent (either by using HTTP/1.1 or sending a `Connection: keep-alive` header), the request processing thread assumes the connection to be idle and passes the connection to the Keep-Alive connection management subsystem.

The Keep-Alive subsystem periodically polls such idle connections and queues those connections with activity into the connection queue for future processing. From there, a request processing thread again retrieves the connection and processes its request. The Keep-Alive subsystem is multi-threaded, as it manages potentially tens of thousands of connections. Efficient polling techniques are used, by dividing the number of connections into smaller

subsets, to determine which connections are ready with requests and which of those connections have idled for sufficient time to deem them closed (beyond a maximum permissible Keep-Alive timeout).

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 Communications 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 `https://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.

Configuring the SIP Service

The SIP service is the component of the Communications Server that provides facilities for deploying SIP applications and for making deployed SIP applications accessible by SIP clients

Unlike the HTTP service, the SIP service does not use virtual servers.

SIP Service

You can change the following attributes of SIP service through the Admin Console, or by using the `asadmin get` and `set` commands. The SIP service attributes apply to all the listeners in the SIP service.

The sub-elements of the SIP service are the Access Log, Request Processing, Keep-Alive, SIP protocol, SIP listeners, and Connection Pool.

To edit SIP service sub-elements and their attributes through the Admin Console, click Configuration > *config-name* > SIP Service.

To list all the sub-elements of the SIP service, you can use the following command:

```
list server-config.sip-service.*
```

Setting the SIP Service Access Log Attributes

You can enable and configure rotation for the access logs for the SIP service. These logs are in the `as-install/domains/domain_dir/logs/sipaccess` directory and are named as follows: `sip_access_log.yyyy-mm-dd`. This location can be changed by editing the `access log` property under the `sip-service` element.

The `accessLoggingEnabled` property under `sip-service` is used for enabling SIP access log. Restart the server after enabling or disabling the access log. Use the following command:

```
asadmin set config.sip-service.property.accessLoggingEnabled=true
```

The `accesslogsize` property under `sip-service` can be used for log rotation based on file size. The default value of this property if set to 2000000.

The following is the list of SIP service access log attributes (The actual attribute name as defined in the `domain.xml` file appears in brackets)

- Rotation (`rotation-enabled`): Check the File Rotation box to enable file rotation.
- Rotation Policy (`rotation-policy`): Choose a policy from the Rotation Policy drop-down list. (The only policy available is `time`).
- Rotation Interval (`rotation-interval-in-minutes`): Enter a numeric value to specify the number of minutes between rotations of the access log. This field is valid only if the Rotation Policy is `time`. The default is 1440 minutes.

When you specify a `rotation-interval`, and the rotation interval has elapsed, the access log file will be rotated only if the resulting rotation suffix of the new access log file is different from the rotation suffix of the existing access log file. So, the rotation interval must coincide with the rotation suffix in order to see your access log file rotated within the expected time-frame interval.

- Rotation Suffix (`rotation-suffix`): Enter a string value to specify the suffix to be added to the log file name after rotation. The default is `YYYY-MM-DD`.
- Format (`format`): Enter a string value to specify the format of the access log. Use the formats shown in the following table. The default format is `%client.name% %auth-user-name% %datetime% %request% %status% %response.length%`.

The following SIP headers are also supported: `client.name`, `auth-user-name`, `datetime`, `request`, `status`, `response.length`, `max.forwards`, `from`, `cseq`, `contact`, `to`, `content.type`, `via`, `call.id`

To enable all headers, run the following command:

```
asadmin set server-config.sip-service.access-log.format="\%client.name%
%auth-user-name% %datetime% %request% %status% %response.length%
%max.forwards% %from% %cseq% %contact% %to% %content.type% %via% %call.id%"
```

Note – Ensure that you escape the format string.

You can edit these properties through the Admin Console or through the following `get` and `set` commands

To get the values of all properties of the SIP service access log, use the following command:
`asadmin get server-config.sip-service.access-log.*`

To set an attribute of the SIP service access log, use the `set` command as follows: `asadmin set server-config.sip-service.access-log.rotation-enabled=false`

Setting the SIP Service Keep Alive Attributes

You can edit these attributes through the Admin Console or through the following get and set commands

To get the values of all attributes and properties of the SIP service keep-alive, use the following command: `asadmin get server-config.sip-service.keep-alive.*`

You can set the `timeout-in-seconds` property of the SIP service keep-alive. By default, this property is set to `-1`. A value of 0 or less means keep alive connections are kept open indefinitely.

To set an attribute of the SIP service keep-alive, use the set command as follows: `asadmin set server-config.sip-service.keep-alive.timeout-in-seconds=0`

Setting the SIP Service Connection Pool Attributes

You can edit these attributes through the Admin Console or through the following get and set commands

To get the values of all attributes of the SIP service connection pool, use the following command: `asadmin get server-config.sip-service.connection-pool.*`

- `Queue size(queue-size-in-bytes)`: Specifies the maximum number of messages that can be queued until threads are available to process them for SIP listeners. The default is `-1` and specifies that there is no limit on the number of messages.
- `Size of Receive Buffer (receive-buffer-size-in-bytes)`: Specifies the size of the byte buffer that is used to read from the socket channel. By default, this property is tuned to the socket channel receive buffer size.
- `Size of Send Buffer (send-buffer-size-in-bytes)`: Specifies the maximum size of the SIP message that can be written out. By default, this property is set to 8192.

To set an attribute of the SIP service connection pool, use the set command as follows: `asadmin set server-config.sip-service.connection-pool.receive-buffer-size-in-bytes=10`

Setting the SIP Service Protocol Attributes

You can edit these attributes through the Admin Console or through the following get and set commands

To get the values of all attributes and properties of the SIP service protocol, use the following command: `asadmin get server-config.sip-service.sip-protocol.*`

- `Enable Error Response (error-response-enabled)`: Specifies whether an error response is sent or not. Set to `false` by default.

- Default TCP Transport(`default-tcp-transport`): If set to true, `transport=tcpi` is inserted in the URI of the contact and record-route headers.

To set an attribute of the SIP service protocol, use the set command as follows: `asadmin set server-config.sip-service.sip-protocol.default-tcp-transport=false`

The SIP Service Protocol has a sub-element called SIP timer. You can list the sub-elements of the SIP service protocol by using the following command: `asadmin list server-config.sip-service.access-log.*`

To get the values of all attributes and properties of the SIP service timers, use the following command: `asadmin get server-config.sip-service.protocol.sip-timers.*`

SIP Listeners

You can configure SIP listeners through the Admin Console or through the get and set commands. The server port on which the SIP container listens on can be configured using a SIP listener.

To list all SIP listeners in the default domain, type the following `asadmin` command:

```
list server.sip-service.sip-listener.*
```

Typical output is as follows:

```
server.sip-service.sip-listener.sip-listener-1
server.sip-service.sip-listener.sip-listener-2
server.sip-service.sip-listener.sip-listener-2.ssl
```

By default, a domain has two SIP listeners created: `sip-listener-1` and `sip-listener-2`, representing the SIP and SIPS ports, respectively. By default, `sip-listener-2` is configured to use TLS transport.

To list all attributes of `sip-listener-1` in the default domain, type the following `asadmin` command:

```
get server.sip-service.sip-listener.sip-listener-1.*
```

Typical output is as follows:

```
server.sip-service.sip-listener.sip-listener-1.enabled = true
server.sip-service.sip-listener.sip-listener-1.id = sip-listener-1
server.sip-service.sip-listener.sip-listener-1.port = 5060
server.sip-service.sip-listener.sip-listener-1.transport = udp_tcp
```

By default, the port is set to 5060 and the address to 0.0.0.0. The 0.0.0.0 address is what we call a BIND to ANY address and which allows you to bind the server socket to any network interface on your system.

To change the default port of `sip-listener-1`, use the following command:

```
set server.sip-service.sip-listener.sip-listener-1.port=5065
```

Communications Server provides the flexibility of changing the SIP listener at runtime. Currently, Communications Server does not consider the effects of those change. It is recommended that you plan ahead of time when you change SIP listener settings.

Creating and Editing a SIP Listener

To create a SIP listener, in the Admin Console, expand the Configuration node, select the configuration you want to configure, then select SIP service, then the SIP Listeners node, and click New. Define the SIP listener settings and click Save.

Alternatively, you can use the `asadmin create-sip-listener` command. A sample command is as follows:

```
asadmin create-sip-listener --simplisteneraddress=127.0.0.1
--simplistenerport=5066 --transport=udp_tcp mysimplistener
```

The following is a list of SIP listener attributes:

- Name (id): Provide a unique identifier for your new listener.
- Network Address (address): Enter 0.0.0.0 if you want the listener to listen on all IP addresses for the server, using a unique port value. Otherwise, type a valid IP address for the server.
- Listener Port (address): Enter a unique port value if the Network Address field is 0.0.0.0, or the desired port value if you are using another IP address.
- Enabled (enabled): Determines whether the listener is active. If set to false, any attempts to connect to the listener result in a socket exception (`java.net.ConnectException`).
- Transport (transport): Specifies the type of transport layer protocol. Allowed values are `udp_tcp` and `tls`.

To change an attribute of the SIP listener you can use the `asadmin set` command or you can use the Admin Console. To edit a SIP listener, in the Admin Console, expand the Configuration node, select the configuration you want to configure, then select SIP service, then the SIP Listeners node, and choose the SIP listener to edit. Edit the SIP listener settings and click Save.

A sample command to change the SIP port of a SIP listener is: `asadmin set server.sip-service.sip-listener.sip-listener-1.port=5065`

▼ To Configure a SIP Listener to Use SSL

- 1 In the Developer profile, expand the Configuration, SIP service, then SIP Listeners node.

- 2 In the Clusters and Enterprise profile, expand the Configuration node, select the configuration you want to configure, then select SIP service, then the SIP Listeners node.
- 3 Select the SIP Listener you want to configure.
- 4 Select the SSL tab to configure the listener to use SSL as follows:
 - Client Authentication: Check this field to force clients to authenticate themselves to the server when using this listener.
 - Certificate NickName: Enter the name of an existing server key pair and certificate. See the Security chapter for more information.
 - SSL3, SSL2, TLS: Check the Enabled checkbox to enable security protocol(s) on the listener. Check either SSL3, SSL2 or TLS, or all three.
 - Ciphersuites: Select the available cipher suite used by the protocol(s) and select the Add button. To enable all cipher suites, select the Add All button.
- 5 Click Save.

More Information Equivalent asadmin command

```
create-ssl
```


Overload Protection

Communications Server provides the overload protection feature to protect systems from overload situations. A typical form of overload protection/load regulation is to reject certain types of requests when the load increases. Depending on the overload situation, certain requests are prioritized and certain requests are rejected to decrease the load in the system. In an overload situation, for instance, Communications Server could reject new SIP sessions so that ongoing sessions are prioritized.

To enable overload protection, set the `olpInserted` property to true. By default, this value is set to false. Use the following command to set this property:

```
asadmin set config.sip-container.property.olpInserted=true
```

Communications Server offers two types of load regulation:

- [“CPU Overload Protection” on page 153](#)
- [“Memory Overload Protection” on page 154](#)

CPU Overload Protection

Communications Server enables you to regulate incoming HTTP and SIP requests based on the CPU load. CPU overload protection can be activated or de-activated both during installation and in runtime.

To enable CPU overload protection, set the `CPUOverLoadRegulation` property to true. By default, this value is set to false. Use the following command to set this property:

```
asadmin set config.sip-container.property.CPUOverLoadRegulation=true
```

You can configure the following threshold levels:

- `HttpThreshold` - All incoming HTTP requests are rejected and 503 responses are sent out, when this threshold is exceeded.

- `IrThreshold` - Initial incoming requests are rejected and 503 responses are sent out when this threshold is exceeded.
- `SrThreshold` - All incoming SIP requests are rejected and 503 responses are sent out, when this threshold is exceeded.
- `MmThreshold` - All incoming HTTP and SIP requests are rejected when this threshold is exceeded.
- `SampleRate` - Interval or periodicity of the measurement of CPU usage.
- `NumberOfSamples` - Number of samples that the CPU usage has to be above the threshold level to trigger an action of either dropping the request or sending 503 responses.

For details about these properties, such as default value and allowed values see [TBDlink](#).

These CPU load regulation properties are applicable only if the `CPUOverLoadRegulation` property needs to be set to true. To view and set these values, use the `asadmin get` and `set` commands. For example, you can use the following command to set the value of the `IrThreshold` property:

```
asadmin set config.sip-container.property.IrThreshold=80
```

Memory Overload Protection

Communications Server enables you to regulate incoming HTTP and SIP requests based on the memory usage. Memory overload protection can be activated or de-activated both during installation and in runtime.

To enable memory overload protection, set the `MemOverLoadRegulation` property to true. By default, this value is set to false. Use the following command to set this property:

```
asadmin set config.sip-container.property.MemOverLoadRegulation=true
```

You can configure the following threshold levels:

- `MemHttpThreshold` - All incoming HTTP requests are rejected and 503 responses are sent out, when this threshold is exceeded.
- `MemIrThreshold` - Initial incoming requests are rejected and 503 responses are sent out when this threshold is exceeded.
- `MemMmThreshold` - All incoming HTTP and SIP requests are rejected when this threshold is exceeded.

Managing Web Services

This chapter describes web services management with Communications Server. Admin Console and the `asadmin` tool enable you to deploy, test, and manage web services. You can quickly visualize, understand, monitor, and manage complex web services. You can see all web services deployed in a domain just as you see Java EE applications and application components such as EJBs.

You can also:

- Track and graph response times and invocation counts for web services in real time.
- Generate alerts on boundary conditions including response time and throughput failures.
- 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 155](#)
- [“Deploying and Testing Web Services” on page 157](#)
- [“Using Web Services Registries” on page 158](#)
- [“Transforming Messages with XSLT Filters” on page 160](#)
- [“Monitoring Web Services” on page 160](#)

Overview of Web Services

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 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.

The eXtensible Markup Language (XML) is a standard developed by the World Wide Web Consortium (W3C) and is one of the foundations on which web services are built. XML enables

web services and clients to communicate with each other in a common language. XML is a simple, flexible, text-based markup language. XML data is marked using tags enclosed in angled brackets. The tags contain the meaning of the data they mark. Such markup allows different systems to easily exchange data with each other.

A Document Type Definition (DTD) or XML Schema Definition (XSD) describes the structure of an XML document. It has information on the tags the corresponding XML document can have, the order of those tags, and so forth.

XSLT, which stands for eXtensible Stylesheet Language Transformation, is used for transforming 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

Universal Description, Discovery, and Integration (UDDI) provides a standard way to register, de-register, and look up web services. Similar to a telephone system's yellow pages, a UDDI registry's 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.

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 specifies 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.

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.

Java EE Web Service Standards

Java APIs for XML processing (JAXP) is a vendor-neutral set of lightweight APIs for parsing or processing XML documents. JAXP enables a web service to “plug in” any conforming XML parser. If no external parser is “plugged in,” then JAXP uses its own XML parser implementation.

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. Similarly, a JAX-RPC based client can interact with a non-Java-based web service implementation.

Java API for XML registries (JAXR), a Java API for accessing business registries, has a flexible architecture that supports UDDI, and other registry specifications (such as ebXML). A JAXR client, which can be a stand-alone Java application or a J2EE component, uses an implementation of the JAXR API provided by a JAXR provider to access business registries. A JAXR provider consists of two parts: a registry-specific JAXR provider, which provides a registry-specific implementation of the API, and a JAXR pluggable provider, which implements those features of the API that are independent of the type of registry. The pluggable provider hides the details of registry-specific providers from clients.

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.

Deploying and Testing Web Services

Communications Server enables you to easily deploy and test web services.

Deploying Web Services

Deploy a web service in an enterprise archive (EAR) just as you would an enterprise application.

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. Communications Server will automatically generate the appropriate web XML files and deploy the web service.

In Admin Console, you can view a list of deployed web services under Application Server > Web Services | General.

Viewing Deployed Web Services

To test a web service with Admin Console, select Applications > 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: Click on the link to display the properties of the web application or enterprise application.
- WSDL: Click on the link to display the WSDL file for the web service.
- Module name: the name of the WAR or EAR file for the web service.
- Mapping File: Click on the link to display the Java WSDL mapping file.
- Webservices.xml: click on the link to display the webservices.xml file.
- Implementation Type: SERVLET or EJB
- Implementation Class Name:
- Deployment Descriptors:

Testing Web Services

Admin Console enables you to test web services and diagnose problems. You can ping a deployed web service with a generic test Servlet. SOAP messages are displayed for each method invocation.

To test a web service with Admin Console, select Applications > Web Services > *web-service-name* | General, then click the Test button.

Web Services Security

Support for SOAP message layer security is based on the SAML token profile of WS-Security. Tamper-proof auditing for Web services is also provided.

Using Web Services Registries

Note – Communications 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 Registry

Add or remove a web services registry with Admin Console at Application Server > Web Services | Registry. Use this page to create a Registry Access Point (RAP). 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 of registry installation>`.
- User name and password for the registry.

The registry JNDI Name is created as a result of the following steps:

- A resource adapter is created that can talk to the registry.
- In the application server context, a JAXR resource adapter comes preconfigured to talk to a UDDI registry. You can also download a SOA registry resource adapter module. The SOA registry is the Sun specific ebXML registry.
- Create a connection resource pool using the resource adapter.
- Create a connector resource using this connection pool.

Publishing a Web Service to a Registry

To publish a web service with Admin Console, select Applications > 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 for this web service. Enter the name of the organization, if you are publishing to a UDDI registry.

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 unpublish the web service, then click Unpublish.

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. All the XSLT files are stored in the `generated/xml/appOrModule` directory of the central repository. These transformation rules are synchronized to the remote server instances.

You can apply transformation rule to a SOAP request or response.

To add a transformation rule to apply to a web service operation with Admin Console, select Applications > Web Services > *web-service-name* | Transformation. Click Add.

A list of transformation rule available for this web service end point is displayed.

Browse to the location of the XSLT file that contains the transformation rule. All the generated XSLT files are stored in the `generated/xml/application or module name/` directory.

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 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.

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. If this transformation rule is applied to a web service endpoint, it is automatically disabled. However, the XSLT file remains in the file path location. Other web service endpoints can use this XSLT file.

Monitoring Web Services

Admin console can track and graphically display 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 Applications > 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, and faults for the web service. Does not perform method-level monitoring.
- **HIGH**- Adds message tracing and monitoring of number of requests per second, average response time, and throughput attributes.

- 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.

Viewing Web Service Statistics

Communications Server provides capabilities to track and graphically display the operational statistics of a web service.

View monitoring statistics at Applications > Web Services > *web-service-name* | Monitor | Statistics. The statistics available are:

- Response time in milli seconds on any successful or unsuccessful operation (Maximum, Minimum, Average).
- Throughput
- Total number of requests
- Total number of faults, including URI of the endpoint where the fault originated
- Total number of authentication failures
- Total number of authorization success

Monitoring Web Service Messages

You can also configure a web service to view messages (default is 25) for a web service endpoint. These messages are stored in the memory of remote server instances. Details of SOAP request, response, and HTTP header information are displayed.

Monitor web service messages at Applications > Web Services > *web-service-name* | Monitor | Messages.

When enabled, you can see the last few (default is 25) messages for a web service end point. These messages are kept in memory of the remote server instances, including details of SOAP requests and responses and HTTP header information.

Displays a list of messages received for the web service. The number of messages displayed depends on the monitoring configuration.

You can also select a filter to view only the success messages or the failure messages.

Configuring the Object Request Broker

This chapter describes how to configure the Object Request Broker (ORB) and IIOP listeners. It has the following sections:

- “An Overview of the Object Request Broker” on page 163
- “Configuring the ORB” on page 164
- “Managing IIOP Listeners” on page 164

An Overview of the Object Request Broker

- “CORBA” on page 163
- “What is the ORB?” on page 164
- “IIOP Listeners” on page 164

CORBA

The Communications 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 Communications 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 Communications Server. For each listener, specify a port number, a network address, and optionally, security attributes.

Configuring the ORB

To configure ORB, in the Admin Console, click the Configuration tab. Click the ORB tab corresponding to the instance you want to configure.

Managing IIOP Listeners

To create, edit and delete IIOP listeners, click the Configuration tab in the Admin Console. Click the ORB tab corresponding to the instance you want to configure. Select the IIOP Listeners tab. For more detailed instructions, consult the Admin Console Online Help.

◆ ◆ ◆ CHAPTER 18

Thread Pools

The Java Virtual Machine (JVM) can support many threads of execution at once. To help performance, the Communications 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 Communications Server, by dividing the Communications Server threads into different thread-pools.

This chapter contains the following topics:

- [“Working with Thread Pools” on page 166](#)

Working with Thread Pools

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 Communications 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 create-threadpool`.

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 Communications Server.

For more details on editing thread pools, click Help in the Admin Console.

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.

You can also create a thread pool from the command-line by using the `asadmin delete-threadpool`.

Configuring ThreadPools for SIP Container

The SIP container uses threads for tasks such as processing an incoming request, send a response, execute timers and initiate new requests with clients. The threads required to perform these tasks are obtained from thread pools that are specifically created and configured for the SIP service.

The Grizzly framework provides a thread pool (called as the pipeline) from which threads are obtained.

The SIP service element provides the following elements for configuring threadpools: `initial-thread-count`, `thread-count`, and `thread-increment`. For complete description of these attributes, see [Sun GlassFish Communications Server 1.5 Administration Reference](#).

Configuring Logging

This chapter briefly describes how to configure logging and view the server log. It contains the following sections:

- “About Logging” on page 169
- “Configuring Logging” on page 172

About Logging

- “Log Records” on page 169
- “The Logger Namespace Hierarchy” on page 170

Log Records

The Communications Server uses the Java EE platform Logging API specified in JSR 047. Communications Server logging messages are recorded in the server log, normally found at *domain-dir/logs/server.log*. When a log is rotated, Communications Server creates a new, empty file named *server.log_date* and renames the old file *server.log_date*, where *date* is the date and time when the file was rotated.

The components of the Communications 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:

```
[#|2006-10-21T13:25:53.852-0400|INFO|sun-appserver9.1|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.
- 2006-10-21T13:25:53.852-0400 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-glassfish-comms-server1.0.
- 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 Communications 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.

The Logger Namespace Hierarchy

The Communications 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 172](#)). The last three modules in the table do not appear on the Log Levels page.

TABLE 19-1 Communications Server Logger Namespaces

Module Name	Namespace
Admin	javax.enterprise.system.tools.admin
ClassLoader	javax.enterprise.system.core.classloading
Configuration	javax.enterprise.system.core.config
Connector	javax.enterprise.resource.resourceadapter
CORBA	javax.enterprise.resource.corba
Deployment	javax.enterprise.system.tools.deployment
EJB Container	javax.enterprise.system.container.ejb

TABLE 19-1 Communications Server Logger Namespaces (Continued)

Module Name	Namespace
Group Management Service (cluster and enterprise profiles only)	javax.ee.enterprise.system.gms
JavaMail	javax.enterprise.resource.javamail
JAXR	javax.enterprise.resource.webservices.registry
JAXRPC	javax.enterprise.resource.webservices.rpc
JAXWS	javax.enterprise.resource.webservices.javaws
JBI	com.sun.jbi
JMS	javax.enterprise.resource.jms
JTA	javax.enterprise.resource.jta
JTS	javax.enterprise.system.core.transaction
MDB Container	javax.enterprise.system.container.ejb.mdb
Naming	javax.enterprise.system.core.naming
Persistence	oracle.toplink.essentials, javax.enterprise.resource.jdo, javax.enterprise.system.container.cmp
Node Agent (cluster and enterprise profiles only)	javax.ee.enterprise.system.nodeagent
Root	javax.enterprise
SAAJ	javax.enterprise.resource.webservices.saaaj
Security	javax.enterprise.system.core.security
Self Management	javax.enterprise.system.core.selfmanagement
Server	javax.enterprise.system
Synchronization (cluster and enterprise profiles only)	javax.ee.enterprise.system.tools.synchronization
Util	javax.enterprise.system.util
Verifier	javax.enterprise.system.tools.verifier
Web Container	javax.enterprise.system.container.web

Configuring Logging

This section contains the following topics:

- [“Configuring General Logging Settings” on page 172](#)
- [“Configuring Log Levels” on page 172](#)
- [“Viewing Server Logs” on page 173](#)

Configuring General Logging Settings

To configure the general logging settings using the Admin Console:

- For the developer profile, go to Application Server → Logging → General
- For the cluster and enterprise profiles, go to Configurations → Configuration → Logging Settings → General

On the General page, enter appropriate values to customize logging to your requirements. Stop and restart the Communications Server.

For details on setting the various configuration parameters, click Help in the Admin Console.

To configure these log settings in `asadmin`, use the `get` and `set` commands.

Configuring Log Levels

To configure log levels using the Admin Console:

- For the developer profile, go to Application Server → Logging → Log Levels
- For the cluster and enterprise profiles, go to Configurations → Configuration → 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 170](#).

For details on setting the various configuration parameters, click Help in the Admin Console.

To configure these log settings in `asadmin`, use the `get` and `set` commands.

Viewing Server Logs

To view the log files:

- In the developer profile, go to Applications Server → Logging → View Log Files.
- In the cluster and enterprise profiles, go to Configurations → Configuration → Logger Settings → General, and click 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 172](#).

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 arrow 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 Communications 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.

Monitoring Components and Services

This chapter contains information about monitoring components using the Communications Server Admin Console. This chapter contains the following sections:

- “About Monitoring” on page 175
- “Enabling and Disabling Monitoring” on page 198
- “Viewing Monitoring Data” on page 200
- “Using JConsole” on page 216

About Monitoring

- “Monitoring in the Communications Server” on page 175
- “Overview of Monitoring” on page 176
- “The Tree Structure of Monitorable Objects” on page 176
- “Statistics for Monitored Components and Services” on page 180

Monitoring in the Communications Server

Use monitoring to observe the runtime state of various components and services deployed in a server instance of the Communications 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.

You can also use management rules to alert you of potential problems with your server, and optionally take action when the server reaches certain performance thresholds. For more information, see [Chapter 21, “Configuring Management Rules.”](#)

Overview of Monitoring

To monitor the Communications 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 [“Enabling and Disabling Monitoring” on page 198](#).

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” on page 200](#).

The Tree Structure of Monitorable Objects

The Communications 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 177](#)
- [“The SIP Service Tree” on page 177](#)
- [“The HTTP Service Tree” on page 178](#)
- [“The Resources Tree” on page 178](#)
- [“The Connector Service Tree” on page 179](#)
- [“The JMS Service Tree” on page 179](#)
- [“The ORB Tree” on page 179](#)
- [“The Thread Pool Tree” on page 180](#)

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 183.

EXAMPLE 20-1 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 SIP Service Tree

The nodes of the SIP service are shown in the following schematic. The nodes at which monitoring information is available are marked with an asterisk (*). See “[SIP Container Statistics](#)” on page 180.

EXAMPLE 20-2 SIP Service Tree Structure

```

sip-service
  |--- network-manager*
  |--- overload-protection-manager*
      |--- session-manager*
      |--- transaction-manager*

```

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 188](#).

EXAMPLE 20-3 HTTP Service Schematic (Developer Profile Version)

```

http-service
  |--- virtual-server-1
      |--- http-listener-1 *
      |--- http-listener-2 *
  |--- virtual-server-2
      |--- http-listener-1 *
      |--- http-listener-2 *

```

EXAMPLE 20-4 HTTP Service Schematic (Cluster and Enterprise Profile Version)

```

http-service *
  |--- connection-queue *
  |--- dns *
  |--- file-cache *
  |--- keep-alive *
  |--- 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 189](#).

EXAMPLE 20-5 Resources Schematic

```

resources
  |--- connection-pool1(either connector-connection-pool or jdbc)*

```

EXAMPLE 20-5 Resources Schematic (Continued)

```
|---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 191.

EXAMPLE 20-6 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 20-7 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 192.

EXAMPLE 20-8 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 193](#).

EXAMPLE 20-9 Thread Pool Schematic

```
thread-pools
  |   |--- thread-pool-1 *
  |   |--- thread-pool-2 *
```

Statistics for Monitored Components and Services

This section describes the monitoring statistics that are available:

- “SIP Container Statistics” on page 180
- “EJB Container Statistics” on page 183
- “Web Container Statistics” on page 187
- “HTTP Service Statistics” on page 188
- “JDBC Connection Pools Statistics” on page 189
- “JMS/Connector Service Statistics” on page 191
- “Statistics for Connection Managers in an ORB” on page 192
- “Thread Pools Statistics” on page 193
- “Transaction Service Statistics” on page 193
- “Java Virtual Machine (JVM) Statistics” on page 194
- “JVM Statistics in Java SE” on page 194

SIP Container Statistics

SIP service statistics are described in the following tables:

The SIP container statistics are described in the following tables:

- Table 20-1
- Table 20-2
- Table 20-3
- Table 20-4

TABLE 20-1 SIP Service Network Manager Statistics

Attribute Name	Data Type	Description
receivedsiprequests	CountStatistic	The number of SIP requests received, including retransmissions.
sentsiprequests	CountStatistic	The number of SIP requests sent, both originated and relayed, including retransmissions and forking.
receivedsipresponses	CountStatistic	The number of SIP responses received, including retransmissions.
sentsipresponses	CountStatistic	The number of SIP responses sent, both originated and relayed, including retransmissions.
invalidsipmessages	CountStatistic	The number of messages received but not recognized as SIP messages, including retransmissions. These messages are not recognized as requests or responses, or these messages are recognized as responses with invalid response code.

TABLE 20-2 Overload Protection Manager Statistics

Attribute Name	Data Type	Description
overloadrejectedsiprequests	CountStatistic	The number of SIP requests rejected due to overload conditions.
overloadrejectedhttprequests	CountStatistic	The number of HTTP requests rejected due to overload conditions.

TABLE 20-3 Session Manager Statistics

Attribute Name	Data Type	Description
concurrentsipdialogs	CountStatistic	The number of SIP dialogs currently stored in the node.

TABLE 20-3 Session Manager Statistics (Continued)

Attribute Name	Data Type	Description
failedsipdialogs	CountStatistic	The number of failed SIP dialog setup attempts. The application or a succeeding node in the signaling path has responded with an 4xx-6xx response code.
expiredsipdialogs	CountStatistic	The number of SIP dialogs that have been removed due to session expiry.
successulsipdialogs	CountStatistic	The number of SIP dialogs that have been successfully established.
totalsipdialogcount	CountStatistic	The total number of SIP dialogs. This count is computed as follows: successulsipdialogs + expiredsipdialogs + failedsipdialogs - concurrentsipdialogs.
totalsipdialoglifetime	CountStatistic	The total lifetime of SIP dialogs in milliseconds.

TABLE 20-4 TransactionManager Statistics

Attribute Name	Data Type	Description
sipservertransactions	CountStatistic	The number of SIP server transactions that are in progress or have reached the terminated state.
sipclienttransactions	CountStatistic	The number of SIP client transactions that are in progress or have reached the terminated state.
totalsiptransactioncount	CountStatistic	The total number of SIP transactions.
totalsiptransactiontime	CountStatistic	The total response time for all SIP transactions in milliseconds.

EJB Container Statistics

The EJB container statistics are described in the following tables:

- [Table 20–5](#)
- [Table 20–6](#)
- [Table 20–7](#)
- [Table 20–8](#)
- [Table 20–9](#)
- [Table 20–10](#)

EJB statistics are described in the following table.

TABLE 20–5 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.
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 20–6 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.

TABLE 20-6 EJB Method Statistics (Continued)

Attribute Name	Data Type	Description
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 20-7 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.

TABLE 20-7 EJB Session Store Statistics (Continued)

Attribute Name	Data Type	Description
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 (enterprise profile only)	CountStatistic	Number of sessions checkpointed using this store.
checkpointSuccessCount (enterprise profile only)	CountStatistic	Number of sessions checkpointed successfully.
checkpointErrorCount (enterprise profile only)	CountStatistic	Number of sessions that couldn't be checkpointed.
checkpointedBeanSize (enterprise profile only)	ValueStatistic	Total number of bytes checkpointed by the store.
checkpointTime enterprise profile only)	TimeStatistic	Time spent on checkpointing beans to the store.

The statistics available for EJB pools are listed in the following table.

TABLE 20-8 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.

TABLE 20-8 EJB Pool Statistics (Continued)

Attribute Name	Data Type	Description
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 20-9 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.
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 20-10 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 177. Web container statistics are displayed for each individual web application. Statistics available for the web container for servlets are shown in [Table 20-11](#), and statistics available for web modules are shown in [Table 20-12](#).

TABLE 20-11 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.
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 187.

TABLE 20-12 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.

TABLE 20-12 Web Container (Web Module) Statistics (Continued)

Statistic	Data Type	Comments
sessionstotal	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.
rejectedsessionstotal	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.
expiredsessionstotal	CountStatistic	Total number of expired sessions for the web module.
sessionsize	AverageRangeStatistic	Size of the session for the web module. Value is either high, low, or average, or is in bytes for serialized sessions.
sessionpersisttime	AverageRangeStatistic	Time (in ms, low, high, or average) taken to persist HTTP session state to back-end store for the web module.
cachedsessionscurrent	CountStatistic	Current number of sessions cached in memory for the web module.
passivatedsessionscurrent	CountStatistic	Current number of sessions passivated for the web module.

HTTP Service Statistics

The statistics available for the HTTP service for the developer profile are shown in the following table.

TABLE 20-13 HTTP Service Statistics (Developer Profile)

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.
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 20-14 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).
numconnfree	Number	CountStatistic	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.

TABLE 20-14 JDBC Connection Pool Statistics (Continued)

Statistic	Units	Data Type	Description
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 [Table 20-15](#). Statistics for Connector Work Management are shown in [Table 20-16](#).

TABLE 20-15 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.

TABLE 20-15 Connector Connection Pool Statistics (Continued)

Statistic	Units	Data Type	Description
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 the following table.

TABLE 20-16 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 Communications 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 the following table.

TABLE 20-17 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.

TABLE 20-17 Connection Manager (in an ORB) Statistics (Continued)

Statistic	Units	Data Type	Description
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 20-18 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.
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 20-19 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.

TABLE 20-19 Transaction Service Statistics (Continued)

Statistic	Data Type	Description
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 20-20 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 Java SE

With Java SE, 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 for Java SE is available in a document titled *Monitoring and Management for the Java Platform*, which is available from <http://java.sun.com/javase/6/docs/technotes/guides/management/>.

The Java SE monitoring tools are discussed at <http://java.sun.com/javase/6/docs/technotes/tools/#manage>.

The statistics available for class loading in the JVM in Java SE are shown in the following table.

TABLE 20-21 JVM Statistics for Java SE- Class Loading

Statistic	Data Type	Description
loadedclasscount	CountStatistic	Number of classes that are currently loaded in the JVM.

TABLE 20–21 JVM Statistics for Java SE- Class Loading (Continued)

Statistic	Data Type	Description
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 Java SE are shown in the following table.

TABLE 20–22 JVM Statistics for Java SE- Compilation

Statistic	Data Type	Description
totalcompilationtime	CountStatistic	Accumulated time (in milliseconds) spent in compilation.

The statistics available for garbage collection in the JVM in Java SE are shown in the following table.

TABLE 20–23 JVM Statistics for Java SE- 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 Java SE are shown in the following table.

TABLE 20–24 JVM Statistics for Java SE- 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.

TABLE 20-24 JVM Statistics for Java SE- Memory (Continued)

Statistic	Data Type	Description
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.
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 Java SE are shown in the following table.

TABLE 20-25 JVM Statistics for Java SE - 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 Java SE are shown in the following table.

TABLE 20-26 JVM Statistics for Java SE - 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.

TABLE 20–26 JVM Statistics for Java SE - Runtime (Continued)

Statistic	Data Type	Description
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.
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 Java SE are shown in the following table.

TABLE 20–27 JVM Statistics for Java SE - 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.

TABLE 20-27 JVM Statistics for Java SE - Thread Info (Continued)

Statistic	Data Type	Description
lockownerid	CountStatistic	ID of the thread that holds the monitor lock of an object on which this thread is blocking.
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 Java SE are shown in the following table.

TABLE 20-28 JVM Statistics for Java SE - 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

This section contains the following topics:

- [“Configuring Monitoring Levels Using the Admin Console” on page 199](#)
- [“To Configure Monitoring Levels Using asadmin” on page 199](#)

Configuring Monitoring Levels Using the Admin Console

To configure monitoring in the Admin Console:

- For the developer profile, go to Configuration → Monitoring
- For the cluster and enterprise profiles, go to Configurations → 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.

▼ To Configure Monitoring Levels Using `asadmin`

- 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.sip-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 200](#)
- [“Viewing Monitoring Data With the asadmin Tool” on page 200](#)

Viewing Monitoring Data in the Admin Console

In the developer profile, to view monitoring data, go to Application Server → Monitor.

In the cluster and enterprise profiles, to view monitoring data for a stand-alone instance, go to Stand-Alone Instances → Instance → Monitor. To view monitoring data for a clustered instance, go to Clusters → Cluster → Instance → Monitor.

You can select and view monitoring data for JVM, server, applications, thread pools, SIP service, HTTP service, transaction service, log statistics, and call flow statistics. See [“The Tree Structure of Monitorable Objects” on page 176](#).

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 [“Statistics for Monitored Components and Services” on page 180](#).

Viewing Monitoring Data With the asadmin Tool

This section contains the following topics:

- [“To Use the asadmin monitor Command to View Monitoring Data” on page 200](#)
- [“To Use the asadmin get and list Commands to View Monitoring Data” on page 201](#)
- [“Understanding and Specifying Dotted Names” on page 202](#)
- [“Examples of the list and get Commands” on page 203](#)
- [“Examples for the list --user admin-user --monitor Command” on page 204](#)
- [“Examples for the get --user admin-user --monitor Command” on page 204](#)
- [“To Use the PetStore Example” on page 206](#)
- [“Expected Output for list and get Commands at All Levels” on page 209](#)

▼ To Use the asadmin monitor Command to View Monitoring Data

asadmin has two ways of viewing monitoring data. The first is to use the `monitor` command. This command prints out the commonly-monitored statistics, and has options for filtering out statistics and capture the output in a Comma Separated Values (CSV) file.

1 To view monitoring data, use the monitor command, and specify the type of monitoring data:

`httplistener, siplistener, keepalive, filecache, connectionqueue, jdbcpool, jvm,`

threadpool, servlet, connection, connectorpool, endpoint, entitybean, messagedriven, statefulsession, statelesssession, httpservice, or webmodule.

For example, to view data for jvm on server, enter the following:

```
asadmin>monitor --type jvm --user adminuser server
```

```

                                JVM Monitoring
UpTime(ms)                                HeapSize(bytes)
current          min          max          low          high          count
327142979       0          531628032  0          45940736  45940736

```

- To view monitoring data and send the output to a CSV file, use the filename option. For example:**

```
asadmin> monitor --type jvm --filename myoutputfile --user adminuser server
```

▼ To Use the asadmin get and list Commands to View Monitoring Data

The monitor command is useful in most situations. However, it does not offer the complete list of all monitorable objects. To view all monitorable data using the asadmin tool, use the asadmin list and asadmin get commands followed by the dotted name of a monitorable object, as follows.

- 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.sip-service
server.thread-pools
```

For further examples using the list command, refer to [“Examples of the list and get Commands” on page 203](#). For further information on the dotted names you can use with the list command, refer to [“Understanding and Specifying Dotted Names” on page 202](#).

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 examples using the `get` command, refer to [“Examples of the list and get Commands” on page 203](#). For further information on the dotted names you can use with the `get` command, refer to [“Understanding and Specifying Dotted Names” on page 202](#).

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-lastssampletime`. The following name addresses the `bytesreceived` attribute:

```
server.http-service.server.http-listener-1.  
bytesreceived-lastssampletime
```

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 209](#).

Examples of the list and get Commands

This section contains the following topics:

- [“Examples for the list --user admin-user --monitor Command”](#) on [page 204](#)
- [“Examples for the get --user admin-user --monitor Command”](#) on [page 204](#)

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 209](#).

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 206](#).

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 209 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 Java EE application:

```
asadmin> get --user admin-user --monitor server.applications.myJavaEEApp.*
```

Returns:

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

There are no monitorable attributes exposed at the Java-EE-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-lastsamplertime
```

Returns:

```
server.jvm.uptime-lastsamplertime = 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 Communications 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 Communications 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=4848
```

- 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 getUser_name 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.getUser_name
```

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-lastsampletime = 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-lastsampletime = 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-lastsampletime = 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 20-29 Top Level

Command	Dotted Name	Output
list -m	server	server.applicationserver.thread-poolserver. resourceserver.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 20-30 Applications Level

Command	Dotted Name	Output
list -m	server.applications or *applications	appl1app2web-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 20-31 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 20-31 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 20-8.

TABLE 20-31 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 20-9.
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 20-6.
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 20-11.

The following table shows the command, dotted name, and corresponding output for the HTTP Service level.

TABLE 20-32 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 20-13 .

The following table shows the command, dotted name, and corresponding output for the thread pools level.

TABLE 20-33 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 20-18 .

The following table shows the command, dotted name, and corresponding output for the resources level.

TABLE 20-34 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 20-14 .

The following table shows the command, dotted name, and corresponding output for the transaction service level.

TABLE 20-35 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 20-19 .

The following table shows the command, dotted name, and corresponding output for the ORB level.

TABLE 20-36 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 20–36 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 20–17 .

The following table shows the command, dotted name, and corresponding output for the JVM level.

TABLE 20–37 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 20–20 .

Using JConsole

This section contains the following topics:

- “Securing JConsole to Application Server Connection” on page 217
- “Prerequisites for Connecting JConsole to Application Server” on page 218
- “Connecting JConsole to Application Server” on page 218
- “Connecting JConsole Securely to Application Server” on page 219

Administration (management and monitoring) of the Communications Server is based on JMX Technology. This means that the managed components are represented as MBeans in the MBeanServer running in the Communications Server's JVM.

Java SE 5 enhances management and monitoring of the JVM by including a Platform MBean Server and by including MBeans to configure the JVM. Communications Server leverages these enhancements and registers its MBeans with the Platform MBean Server. Thus a JMX Connector Client gets a unified view of JVM MBeans as well as Communications Server MBeans.

To view all the MBeans, Communications Server provides a configuration of the Standard JMX Connector Server called System JMX Connector Server. As part of Communications Server startup, an instance of this JMX Connector Server is started. Any compliant JMX connector client can connect to the server using this Connector Server.

Java SE also provides tools to connect to an MBean Server and view MBeans registered with it. JConsole is one such popular JMX Connector Client and is available as part of the standard Java SE distribution. For more information on JConsole, see <http://java.sun.com/javase/6/docs/technotes/guides/management/jconsole.html>

When you configure JConsole with Communications Server, Communications 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 218 shows how to make a successful connection .

Securing JConsole to Application Server Connection

There are subtle differences in how to connect to Communications 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, the developer profile of Communications Server is configured with a non-secure System JMX Connector Server.
- By default, cluster and enterprise profiles of Communications Server are configured with a secure System JMX Connector Server.
- The protocol used for communication is RMI/JRMP. If security is enabled for the JMX Connector, the protocol used is RMI/JRMP over SSL.

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 developer profile domain on a machine such as `appserver.sun.com`, you will see the following in the Domain Administration Server (DAS) `domain.xml` file:

```
<!-- 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 cluster or enterprise profile, or if you have turned on security for the JMX Connector in the developer profile, 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. For this example, the Communications 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 Communications Server. Let us assume that the client end is a Windows machine with Java SE 6.0 and Communications Server installed.

Note – The Communications Server installation is needed on the client end only when your Communications Server domain has security enabled on the remote machine (the default for cluster and enterprise profiles). If you just want to administer an Communications Server developer profile domain on the Solaris machine above, you do not need the Communications 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 Communications Server without security enabled on the JMX Connector. By default, security is not enabled on Communications Server for the developer profile.

- 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 administration user name and password refers to the administration 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 Communications Server with security enabled on the JMX Connector. By default, security is enabled on Communications Server cluster or enterprise profiles. Use this procedure if you have security enabled on the developer profile's JMX Connector.

- 1 **Install Communications 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 Communications Server.

- 2 **Start the Communications Server on `appserver.sun.com`.**

Since this is a cluster or enterprise domain, the system JMX Connector server is secure. To enable security on the developer profile JMX Connector, see the Admin Console online help.

- 3 **From the local Communications Server installation, run `install-dir\bin\asadmin list --user admin --secure=true --host appserver.sun.com --port 4848` (where 4848 is the server's administration port).**

Though `asadmin list` command is chosen for this example, you can run any remote `asadmin` command. You are prompted to accept the certificate sent by the DAS of `appserver.sun.com`.

- 4 **Press `y` to accept the certificate sent by the DAS 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 administration user name and password refers to the administration 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 conditions and improve availability by preventing failures. This section also contains information on the self-management templates, which are predefined management rules that you can customize.

This section contains the following topics:

- [“About Management Rules” on page 221](#)
- [“Configuring Management Rules” on page 222](#)

About Management Rules

Management rules enable you to automate routine administration tasks, configure self-tuning of the application server for diverse runtime condition and improve availability by preventing failures. A management rule contains an action to be taken when a specified event occurs or a set threshold is reached. You can set management rules that can automatically take corrective action, based on events that you specify.

A management rule consists of two parts — event and action:

- An *event* uses the JMX notification mechanism to trigger a predefined action.
- An *action* is triggered when an associated event occurs. An action is an MBean which is a notification listener which implements `javax.management.NotificationListener`.

For example, an event could be a SEVERE message logged by the EJB logger, and an action could be alerting an administrator with the log message contents. When the event happens, event data is passed as part of `userData` part of the `javax.management.Notification`.

The action specified in your rule has to be implemented as a custom 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 GlassFish Communications Server 1.5 Developer’s Guide*.

The Communications Server provides some useful events, which you can further extend by writing custom MBeans to emit notifications. Each event can be further customized by changing its properties.

The following event types are available:

- **Monitor events:** Monitors an attribute of an MBean. Monitor events have similar capabilities to `javax.management.monitor` package capabilities. In addition to monitoring simple attributes, as Java SE 5 `javax.management.monitor` does, monitor events also support monitoring complex attributes.
- **Notification events:** Notifies of events from a custom MBean. Use these events to write custom events and thus extend the event dictionary. Any MBeans which can emit a notification can be an event.
- **System events:**
 - **Lifecycle:** Events for sever startup, shutdown, and termination.
 - **Log:** Events triggered when the specified logger writes a log entry. For example, you could create a management rule to send an alert to an administrator when an EJB container logger logs a SEVERE log entry.
 - **Timer:** Events triggered at the specified date and time, at the specified interval, and so on. These events have capabilities similar to the `javax.management.timer` package.
 - **Trace:** Events triggered on Entry and Exit of HTTP/IIOP request methods, EJB methods, and Web methods. For example, you can design a servlet filter used to log interactions with a servlet as a management rule using Web method Entry and Exit events.
 - **Cluster:** Events triggered when a cluster or instance starts, stops, or fails. These events use the Group Management System cluster monitoring.

Configuring Management Rules

To configure management rules in the Admin Console:

- In the developer profile, go to Configuration → Management Rules
- In the cluster and enterprise profiles, go to Configurations → Configuration → Management Rules

Note – On this page, check All Rules Enabled to enable management rules globally. If management rules are not enabled globally, none of the management rules are executed.

In addition, to enable a individual management rule, you must enable the rule on this page by clicking the box next to the rule and clicking Enable.

A rule's MBeans must also be enabled on a target. To enable MBeans, go to Custom MBeans → MBean. On the Edit Custom MBean page, click the Target tab to access the Custom MBean Targets page, where you can enable the MBeans on some or all of the targets.

For details, see the online help.

To create management rules from the command line, use the `create-management-rule` command. To set properties of the management rules, use the `get` and `set` commands. To list and delete management rules, use `list-management-rules` and `delete-management-rule`. For more information, see the online help for these commands, or [Sun GlassFish Communications Server 1.5 Reference Manual](#).

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 Communications Server, being a Java process, requires a JVM in order to run and support the Java applications running on it. JVM settings are part of an application server configuration.

This chapter explains how to configure the Java Virtual Machine (JVM™) and other advanced settings. It contains the following sections:

- “[Tuning the JVM Settings](#)” on page 225
- “[Configuring Advanced Settings](#)” on page 226

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 Communications Server relies on the Java SE software.

Note – If you enter a nonexistent directory name or the installation directory name of an unsupported version of the Java EE software, then the Communications Server will not start.

- **Javac Options:** Enter the command-line options for the Java programming language compiler. The Communications 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.

- **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 Communications 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.

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 or Node Agent

If your domain or node agent is stopped unexpectedly (for example, if you need to restart your machine), you can configure your system to automatically restart the domain or node agent.

This Appendix contains the following topics:

- “Restarting Automatically on Solaris 10” on page 227
- “Restarting Automatically Using inittab on Solaris 9 and Linux Platforms” on page 229
- “Restarting Automatically on the Microsoft Windows Platform” on page 229
- “Security for Automatic Restarts” on page 231

Restarting Automatically on Solaris 10

Solaris 10 users can use the command `asadmin create-service` to create a service that restarts a node agent or Domain Administration Server (DAS). The service created uses the Solaris Service Management Facility (SMF).

The process that a service starts depends on whether the service is to restart a DAS or a node agent.

- If the service is to restart a DAS, the process is `asadmin start-domain`.
- If the service is to restart a node agent, the process is `asadmin start-node-agent`.

The service grants to the process the privileges of the user that runs the process. When you use the command `asadmin create-service` to create an SMF service, the default user is the superuser. If you require a different user to run the process, specify the user in `method_credential`.

If your process is to bind to a privileged port of the Solaris OS, the process requires the `net_privaddr` privilege. The privileged ports of the Solaris OS have a port numbers less than 1024.

To determine if a user has the `net_privaddr` privilege, log in as that user and type the command `ppriv -l | grep net_privaddr`.

To run the `asadmin create-service` command, you must have `solaris.smf.*` authorization. See the `useradd` and `usermod` man pages to find out how to set the authorizations. You must also have write permission in the directory tree:

`/var/svc/manifest/application/SUNWappserver`. Usually, the superuser has both these permissions. Additionally, the Solaris 10 administration commands such as `svccfg`, `svcs`, and `auths` must be available in the `PATH`. For complete information on running this command, see [create-service\(1\)](#).

The syntax is as follows:

```
asadmin create-service [--name service-name] [--type das|node-agent]
--passwordfile password-file [--serviceproperties serviceproperties]
domain-or-node-agent-configuration-directory
```

For example, to create a service called `domain1` for `domain1`:

1. Run the following:

```
asadmin create-service --type das --passwordfile password.txt
/appserver/domains/domain1
```

This creates a service to restart the domain `domain1` automatically. In the background, the command creates a manifest file from a template, validates the file, and imports it as a service.

Note – If a particular Communications Server domain should not have default user privileges, modify the service's manifest and reimport the service. To determine a user's privileges, log in as that user and type the command `ppriv -l`.

2. Once the service is created, enable it using the `svcadm enable` command:

```
svcadm enable /appserver/domains/domain1
```

3. Once enabled, if the domain goes down, SMF restarts it.

As you administer your service, the following Solaris commands are useful:

- `auths`
- `smf_security`
- `svcadm`
- `svccfg`
- `rbac`
- `useradd`
- `usermod`

For more information on these commands, see the command manpages.

Restarting Automatically Using inittab on Solaris 9 and Linux Platforms

To restart your domain on the Solaris 9 or Linux 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 Communications 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.

To restart a node agent, the syntax is similar. For example, to restart `agent1` for an Communications Server installed in the `opt/SUNWappserver` directory, using a password file called `password.txt`:

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-node-agent --user admin
--passwordfile /opt/SUNWappserver/password.txt agent1
```

Restarting Automatically on the Microsoft Windows Platform

To restart automatically on Microsoft Windows, create a Windows Service and prevent the service from shutting down when a user logs out.

Creating a Windows Service

Use the `appservService.exe` and `appserverAgentService.exe` executable files shipped with the Sun GlassFish Communications Server in conjunction with the Service Control command (`sc.exe`) provided by Microsoft.

The `sc.exe` command comes with Windows XP and is in the `system32` subdirectory of the Windows installation directory (usually either `C:\windows\system32` or `C:\winnt\system32`). As of this writing, the Windows 2000 `sc.exe` is available for download at <ftp://ftp.microsoft.com/reskit/win2000/sc.zip>. For more information on using `sc.exe`, see http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndllpro/html/msdn_scmslite.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"
```

Note – There is no space between binpath and the equals sign (=). There must be a space after the equals sign and before the path.

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"
```

To create a service that starts and stops the node agent agent1:

```
C:\windows\system32\sc.exe create agent1 binPath=
"C:\Sun\AppServer\lib\appservAgentService.exe \"C:\Sun\AppServer\bin\asadmin.bat
start-node-agent --user admin --passwordfile C:\Sun\AppServer\password.txt agent1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-node-agent agent1\" start= auto
DisplayName= "SunJavaSystemAppServer AGENT1"
```

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 or node agent, 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.

If your sc.exe create command did not properly create the service, delete the service and try again. To delete the service, use the sc.exe delete "*service-name*" command.

Preventing the Service From Shutting Down When a User Logs Out

By default, the Java VM catches signals from Windows that indicate that the operating system is shutting down, or that a user is logging out, and shuts itself down cleanly. This behavior causes the Communications Server service to shut down when a user logs out of Windows. To prevent the service from shutting down when a user logs out, set the `-Xrs` Java VM option (<http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/java.html#Xrs>).

To set the `-Xrs` Java VM option, add the following line to the section of the `as-install\domains\domain-name\config\domain.xml` file that defines Java VM options:

```
<jvm-options>-Xrs</jvm-options>
```

If the Communications Server service is running, stop and restart the service for your changes to become effective.

Note – In some Windows 2003 Server installations, adding the `-Xrs` option to the `domain.xml` file fails to prevent the service from shutting down. In this situation, add the option to the `as-install\lib\processLauncher.xml` file as follows:

```
<process name="as-service-name">
  ...
  <sysproperty key="-Xrs"/>
  ...
</process>
```

Security for Automatic Restarts

If you are using the cluster or enterprise profile, the administration password and master password are required when automatically restarting Communications Server. If you are using the Developer Profile, no

Handle the password and master password requirements for cluster and enterprise profiles 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.

For example:

1. Create domain with a saved master password. In this syntax, you are prompted for the administration 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= "SunJavaSystemAppServer 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 `init` tab 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
```


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 233](#)
- [“Elements Not Aliased” on page 235](#)

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
<code>applications</code>	<code>domain.applications</code>
<code>resources</code>	<code>domain.resources</code>
<code>configurations</code>	<code>domain.configs</code>
<code>servers</code>	<code>domain.servers</code> 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 Applicable only for Sun GlassFish Enterprise Server
system-property	domain.system-property
converged-lb-configs	domain.converged-lb-configs Applicable only for Sun GlassFish Communications Server

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
<code>target.applications.*</code>	<code>domain.applications.*</code>	The alias resolves to applications referenced by the <i>target</i> only.
<code>target.resources.*</code>	<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 some of the 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. For a complete list, use the `asadmin list target.*` command.

Dotted Name	Aliased to
<code>target.http-service</code>	<code>config-name.http-service</code>
<code>target.iiop-service</code>	<code>config-name.iiop-service</code>

Dotted Name	Aliased to
<i>target.admin-service</i>	<i>config-name.admin-service</i>
<i>target.web-container</i>	<i>config-name.web-container</i>
<i>target.sip-container</i>	<i>config-name.sip-container</i> Applicable only for Sun GlassFish Communications Server
<i>target.ejb-container</i>	<i>config-name.ejb-container</i>
<i>target.mdb-container</i>	<i>config-name.mdb-container</i>
<i>target.jms-service</i>	<i>config-name.jms-service</i>
<i>target.sip-service</i>	<i>config-name.sip-service</i> Applicable only for Sun GlassFish Communications Server
<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, manage users, resources, and applications.

This chapter contains the following sections:

- “Common Options for Remote Commands” on page 240
- “The Multimode Command” on page 241
- “The Get, Set, and List Commands” on page 242
- “Server Lifecycle Commands” on page 243
- “List and Status Commands” on page 244
- “Deployment Commands” on page 245
- “Version Commands” on page 246
- “Message Queue Administration Commands” on page 246
- “Resource Management Commands” on page 247
- “Configuration Commands” on page 249
- “User Management Commands” on page 253
- “Rules and Monitoring Commands” on page 253
- “Database Commands” on page 254
- “Diagnostic and Logging Commands” on page 254
- “Web Service Commands” on page 255
- “Security Service Commands” on page 255
- “Password Commands” on page 256
- “Verify Command” on page 257
- “Custom MBean Commands” on page 257
- “Service Command” on page 258
- “Property Command” on page 258

The `asadmin` Utility

Use the `asadmin` utility to perform any administrative tasks for the Application Server. You can use this `asadmin` utility in place of using the Administrator interface.

The `asadmin` utility invokes subcommands that identify the operation or task you wish to perform. Subcommands are case-sensitive. Short option arguments have a single dash (-); while long option arguments have two dashes (--). Options control how the utility performs a subcommand. 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.

`asadmin` can be used in command shell invocation or multi command mode (known as `multimode`). 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 options. By default, the interactive option is enabled. It prompts you for the required arguments. You can use the interactive option in command shell invocation under all circumstances. You can use the interactive option in `multimode` when you run one subcommand at a time from the command prompt; and when you run in `multimode` from a file. Subcommands in `multimode`, when piped from an input stream, and subcommands invoked from another program, cannot run in the interactive option.

Local subcommands 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 subcommand and have access (permissions) for the installation and domain directories. Remote subcommands are always executed by connecting to an administration server and executing the subcommand there. A running administration server is required. All remote subcommands require the following options:

- `-u` --user authorized domain application server administrative username.
- `--passwordfile` the file containing the domain application server password in the following form: `AS_ADMIN_PASSWORD=password`. Where *password* is the actual administrator password.
- `-H` --host machine name where the domain application server is running.
- `-p` --port port number of the domain application server listening for administration requests. The default port number for Platform Edition is 4848.

- `-s` `--secure` if true, uses SSL/TLS to communicate with the domain application server.
- `-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.
- `-e` `--echo` setting to true will echo the command line statement on the standard output. Default is false.
- `-I` `--interactive` if set to true (default), only the required password options are prompted.
- `-h` `--help` displays the help text for the command.

For subcommands 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 subcommand will run in remote mode. Additionally, for subcommands that can be executed locally or remotely, if the `--local` option is set to true, the subcommand will run locally. Also, if none of the local options are set, either on the command line or in the environment, the subcommand is executed locally by default. Setting the `--local` option to true overrides the local `--host`, `--port`, `--user`, and `--passwordfile` settings, even if specified. The subcommand will run in local mode.

Subcommands that can be executed locally accept the `--domain` option to specify the domain of interest which assumes the domain as the default domain if there is only one. If there is more than one domain, the `--domain` option is a required option. For subcommands that can be run locally or remotely, when run remotely with the `--host`, `--port`, `--user`, and `--passwordfile` options specified, the `--domain` option is ignored. The `--domain` option is ignored if the subcommand will be run in remote mode. Note that there is one administration instance per domain, so on a single machine with multiple domains, local execution must specify the domain, and remote execution must specify the `--host`, `--port`, `--user`, and `--passwordfile` options for the administration instance for that domain.

For security purposes, you can set the password for a subcommand 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 C-1 Passwordfile contents

```
AS_ADMIN_PASSWORD=value
AS_ADMIN_ADMINPASSWORD=value
AS_ADMIN_USERPASSWORD=value
AS_ADMIN_MASTERPASSWORD=value
```

If **AS_ADMIN_PASSWORD** has been exported to the global environment, specifying the `--passwordfile` option will produce a warning about using the `--password` option. Unset **AS_ADMIN_PASSWORD** to prevent this from happening. The master password is not propagated on the command line or an environment variable, but can be specified in the passwordfile.

To use the `--secure` option, you must use the `set` command to enable the security `--enabled` flag in the `admin http-listener` in the `domain.xml`. When you use the `asadmin` subcommands to create and/or delete, you must restart the server for the newly created command to take affect. Use the `start-domain` command to restart the server.

To access the manpages for the Application Server command-line interface subcommands 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 subcommands by invoking the `--help` option. If you specify a subcommand, the usage information for that subcommand is displayed. Using the `--help` option without a subcommand displays a listing of all the available subcommands.

Common Options for Remote Commands

All the remote commands require the following common options:

TABLE C-1 Remote Commands Required Options

Option	Definition
<code>--host</code>	The machine name where the domain administration server is running. The default value is <code>localhost</code> .
<code>--port</code>	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.
<code>--user</code>	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.

TABLE C-1 Remote Commands Required Options (Continued)

Option	Definition
<code>--passwordfile</code>	<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 <code>password</code> is the actual administrator password. Other passwords that can be specified include <code>AS_ADMIN_MAPPEDPASSWORD</code>, <code>AS_ADMIN_USERPASSWORD</code>, and <code>AS_ADMIN_ALIASPASSWORD</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>
<code>--secure</code>	If set to true, uses SSL/TLS to communicate with the domain administration server.
<code>--interactive</code>	If set to true (default), only the required password options are prompted.
<code>--terse</code>	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.
<code>--echo</code>	Setting to true will echo the command line statement on the standard output. Default is false.
<code>--help</code>	Displays the help text for the command.

The 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 *multimodesession*; once you exit the second *multimode* environment, you return to your original *multimode* environment.

The Get, Set, and List Commands

The `asadmin` `get`, `set` and `list` commands work in tandem to provide a navigation mechanism for the Application Server's abstract 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 `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. See the examples for further clarification of the possible navigation of the hierarchies and management components.

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.
- 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 abstract hierarchy. Upon providing this name to `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` will return 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 `appserver` are: `* list "*" and <command>* list * --monitor`. Consult the `get` or `list` commands manpages 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, or an instance.

TABLE C-2 Server Lifecycle Commands

Command	Definition
<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 application server. A domain can exist independent of other domains. Any user who has access to the <code>asadmin</code> script 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 C-2 Server Lifecycle Commands (Continued)

Command	Definition
<code>start-domain</code>	Starts a domain. If the domain directory is not specified, the domain in the default <code>install_dir/domains</code> directory is started. If there are two or more domains, the <code>domain_name</code> operand must be specified.
<code>stop-domain</code>	Stops the Domain Administration Server of the specified domain.
<code>restore-domain</code>	Restores files under the domain from a backup directory.
<code>list-domains</code>	Lists the domain. If the domain directory is not specified, the domain in the default <code>install_dir/domains</code> directory is listed. If there is more than one domain, the <code>domain_name</code> operand must be specified.
<code>backup-domain</code>	Backs up files under the named domain.
<code>login</code>	Lets you log in to 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 during the <code>create-domain</code> command while using the <code>--saveLogin</code> option. The domain must be running for this command to run.
<code>create-instance</code>	Creates a new server instance residing on a local or remote machine.
<code>delete-instance</code>	Deletes the server instance. This command can be run remotely or locally. The user authenticates using the password identified for the administration server. Additionally, the instance must already exist within the domain served by the administration server. Use this command with discretion since it is destructive and cannot be undone.

List and Status Commands

The list and status commands display the status of a deployed component.

TABLE C-3 List and Status Commands

Command	Definition
<code>show-component-status</code>	Gets the status of the deployed component. The status is a string representation returned by the server. The possible status strings include status of <code>app-name</code> is enabled or status of <code>app-name</code> is disabled.

TABLE C-3 List and Status Commands (Continued)

<code>list-components</code>	Lists all deployed Java EE 5 components. If the <code>-type</code> option is not specified, all components are listed.
<code>list-sub-components</code>	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.
<code>enable</code>	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.
<code>disable</code>	Immediately disables the named component. The component must have been deployed. If the component has not been deployed, an error message is returned.
<code>export</code>	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.
<code>get</code>	Gets the names and values of attributes.
<code>set</code>	Sets the values of one or more configurable attribute.
<code>list</code>	Lists the configurable element. On Solaris, quotes are needed when executing commands with <code>*</code> as the option value or operand.
<code>unset</code>	Removes one or more variables you set for the multimode environment. The variables and their associated values will no longer exist in the environment.

Deployment Commands

The deployment commands deploy an application or get the client stubs.

TABLE C-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.

Version Commands

The version commands return the version string, display a list of all the `asadmin` commands, and allow you to install the license file.

TABLE C-5 Version Commands

Command	Definition
<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.
<code>shutdown</code>	Gracefully brings down the administration server and all the running instances. You must manually start the administration server to bring it up again.

Message Queue Administration Commands

The Message Queue administration commands allow you to manage the JMS destinations.

TABLE C-6 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 C-7 Resource Management Commands

Command	Definition
<code>create-jdbc-connection-pool</code>	Registers a new JDBC connection pool with the specified JDBC connection pool name.
<code>delete-jdbc-connection-pool</code>	Deletes a JDBC connection pool. The operand identifies the JDBC connection pool to be deleted.
<code>list-jdbc-connection-pools</code>	Gets the JDBC connection pools that have been created.
<code>create-jdbc-resource</code>	Creates a new JDBC resource.
<code>delete-jdbc-resource</code>	Removes a JDBC resource with the specified JNDI name.
<code>list-jdbc-resources</code>	Displays a list of JDBC resources that have been created.
<code>create-jms-resource</code>	Creates a Java Message Service (JMS) connection factory resource or a JMS destination resource.
<code>delete-jms-resource</code>	Removes the specified JMS resource.
<code>list-jms-resources</code>	Lists the existing JMS resources (destination and connection factory resources).
<code>create-jndi-resource</code>	Registers a JNDI resource.
<code>delete-jndi-resource</code>	Removes the JNDI resource with the specified JNDI name.
<code>list-jndi-resources</code>	Identifies all the existing JNDI resources.
<code>list-jndi-entries</code>	Browses and queries the JNDI tree.
<code>create-javamail-resource</code>	Creates a JavaMail session resource.
<code>delete-javamail-resource</code>	Removes the specified JavaMail session resource.
<code>list-javamail-resources</code>	Lists the existing JavaMail session resources.
<code>create-persistence-resource</code>	Registers a persistence resource.
<code>delete-persistence-resource</code>	Removes a persistence resource. When you delete a persistence resource, the command also removes the JDBC resource if it was created using the <code>create-persistence-resource</code> command.
<code>list-persistence-resources</code>	Displays all the persistence resources.
<code>create-custom-resource</code>	Creates a custom resource. A custom resource specifies a custom server-wide resource object factory that implements the <code>javax.naming.spi.ObjectFactory</code> interface.

TABLE C-7 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>	Creates the administered object that has a 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 enterprise server or database is started.

Configuration Commands

The configuration commands allow you to construct IIOP listeners, lifecycle modules, HTTP and IIOP listeners, profilers, and other subsystems.

This section contains the following topics:

- “SIP Listener Commands” on page 249
- “HTTP and IIOP Listener Commands” on page 249
- “Lifecycle and Audit Module Commands” on page 250
- “Profiler and SSL Commands” on page 250
- “JVM Options and Virtual Server Commands” on page 251
- “Threadpool and Auth-Realm Commands” on page 251
- “Transaction and Timer Commands” on page 252

SIP Listener Commands

The SIP listener commands help you manage listeners. These commands are supported in remote mode only.

TABLE C-8 SIP Listener Commands

Command	Definition
<code>create-sip-listener</code>	Adds a new SIP listener socket.
<code>delete-sip-listener</code>	Removes the specified SIP listener.
<code>list-sip-listeners</code>	Lists the existing SIP listeners.

HTTP and IIOP Listener Commands

The HTTP and IIOP listener commands help you manage listeners. These commands are supported in remote mode only.

TABLE C-9 IIOP Listener Commands

Command	Definition
<code>create-http-listener</code>	Adds a new HTTP listener socket.
<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.

TABLE C-9 IIOp Listener Commands (Continued)

Command	Definition
<code>delete-iioP-listener</code>	Removes the specified IIOp listener.
<code>list-iioP-listeners</code>	Lists the existing IIOp listeners.

Lifecycle and Audit Module Commands

The lifecycle and audit module commands help you control lifecycle modules and optional plugin modules which implement audit capabilities. The commands are supported in remote mode only.

TABLE C-10 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 SSL Commands

The Profiler and SSL commands allow you to administrate profilers and SSL client configurations. These commands are supported in remote mode only.

TABLE C-11 Profiler and SSL Commands

Command	Definition
<code>create-profiler</code>	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.

TABLE C-11 Profiler and SSL Commands (Continued)

Command	Definition
<code>delete-profiler</code>	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.
<code>create-ssl</code>	Creates and configures the SSL element in the selected SIP/ HTTP/ IIOP listener, or IIOP service to enable secure communication on that listener/service.
<code>delete-ssl</code>	Deletes the SSL element in the selected SIP/ HTTP/ IIOP listener, or IIOP service.

JVM Options and Virtual Server Commands

The JVM options and Virtual Server commands allow you to control these elements. These commands are supported in remote mode only.

TABLE C-12 JVM Options and Virtual Server Commands

Command	Definition
<code>create-jvm-option</code>	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.
<code>delete-jvm-option</code>	Removes JVM options from the Java configuration or profiler elements of the <code>domain.xml</code> file.
<code>create-virtual-server</code>	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.
<code>delete-virtual-server</code>	Removes the virtual server with the specified virtual server ID.

Threadpool and Auth-Realm Commands

The threadpool and auth-realm commands allow you to control these elements. These commands are supported in remote mode only.

TABLE C-13 Threadpool and Auth-Realm 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.
<code>create-auth-realm</code>	Adds the named authentication realm.
<code>delete-auth-realm</code>	Removes the named authentication realm.

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 C-14 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>unpublish-from-registry</code>	
<code>list-timers</code>	Lists the timers owned by a specific server instance

Registry Commands

The registry commands allow you to publish or unpublish webservice artifacts.

TABLE C-15 Transaction Commands

Command	Definition
<code>publish-to-registry</code>	Publishes the web service artifacts to registries.
<code>unpublish-from-registry</code>	Unpublishes the web service artifacts from the registries.
<code>list-registry-locations</code>	

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 C-16 User Management Commands

Command	Definition
<code>create-file-user</code>	Creates an entry in the keyfile with the specified username, password, and groups. Multiple groups can be created by separating them with a colon (:).
<code>delete-file-user</code>	Deletes the entry in the keyfile with the specified username.
<code>update-file-user</code>	Updates an existing entry in the keyfile using the specified <code>user_name</code> , <code>user_password</code> and groups. Multiple groups can be entered by separating them, with a colon (:).
<code>list-file-users</code>	Creates a list of file users supported by file realm authentication.
<code>list-file-groups</code>	Administers file users and groups supported by the file realm authentication. This command lists available groups in the file user.

Rules and Monitoring Commands

Rules and monitoring commands allow you to manage rules and monitor the server. These commands are supported in remote mode only.

TABLE C-17 Rules and Monitoring Commands

Command	Definition
<code>create-management-rule</code>	Creates a new management rule to intelligently self-manage the application server installation and deployed applications.
<code>delete-management-rule</code>	Removes the management rule you specify.

TABLE C-17 Rules and Monitoring Commands (Continued)

Command	Definition
<code>create-transformation-rule</code>	Creates an XSLT transformation rule that can be applied to a webservice operation. The rule can be applied either to a request or to a response.
<code>delete-transformation-rule</code>	Deletes an XSLT transformation rule of a given web service.
<code>start-callflow-monitoring</code>	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 <code>callflow-monitoring</code> is ON.
<code>stop-callflow-monitoring</code>	Disables collection of call flow information of a request.

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 C-18 Database Commands

Command	Definition
<code>start-database</code>	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.
<code>stop-database</code>	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 C-19 Diagnostic and Logging Commands

Command	Definition
<code>generate-diagnostic-report</code>	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.
<code>display-error-statistics</code>	Displays a summary list of severities and warnings in <code>server.log</code> since the last server restart.

TABLE C-19 Diagnostic and Logging Commands (Continued)

Command	Definition
display-error-distribution	Displays distribution of errors from instance server .log at module level.
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 C-20 Web Service Commands

Command	Definition
configure-webservice-management	configure the monitoring or the maxhistory attributes of a deployed web service.
create-transformation-rule	Creates an XSLT transformation rule that can be applied to a web service operation. The rule can be applied either to a request or to a response.
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 registries.
unpublish-from-registry	Unpublishes the web service artifacts from the registries.
list-registry-locations	Displays a list of configured web service registry 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 C-21 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-map</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>).

Password Commands

The password commands allow you to manage passwords and ensure security for the application server.

TABLE C-22 Password Commands

Command	Definition
<code>create-password-alias</code>	Creates an alias for a password and stores it in <code>domain.xml</code> . An alias is a token of the form <code>\${ALIAS=password-alias-password}</code> . 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.
<code>delete-password-alias</code>	Deletes a password alias.
<code>update-password-alias</code>	Updates the password alias IDs in the named target.
<code>list-password-aliases</code>	Lists all password aliases.
<code>change-admin-password</code>	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).
<code>change-master-password</code>	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 Command

The XML verifier command verifies the content of the `domain.xml` file.

TABLE C-23 Verify Command

Command	Definition
<code>verify-domain.xml</code>	Verifies the content of the <code>domain.xml</code> file.

Custom MBean Commands

The MBean commands allow you to manage and register custom MBeans. The commands are supported in remote mode only.

TABLE C-24 Custom MBean Commands

Command	Definition
<code>create-mbean</code>	Creates and registers a custom MBean. If the target MBeanServer is not running, the MBean is not registered.

TABLE C-24 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.

Service Command

The service command allows you to configure the starting of the Domain Administration Server (DAS).

TABLE C-25 Service Command

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.

Property Command

Shared server instances will often need to override attributes defined in their referenced configuration. Any configuration attribute in a server instance can be overridden through a system property of the corresponding name. Use the system property commands to manage these shared server instances.

TABLE C-26 Property Command

Command	Definition
<code>create-system-property</code>	Creates one system property of the domain, configuration, or server instance, at a time.
<code>delete-system-property</code>	Removes one system property of a domain, configuration, or server instance.
<code>list-system-properties</code>	Displays the system properties of a domain, configuration, or server instance.

Index

A

ACC

See containers

application client, 95

acceptor threads, in HTTP listeners, 145

Admin Console, 26

applets, 95

asadmin utility, 27

B

bean-cache, monitoring attribute names, 186

Binding Components, overview, 42

C

cache-hits, 186

cache-misses, 186

CloudScape Type 4 JDBC driver, 63

connection factories, JMS, overview, 65-66

connector connection pools, JMS resources and, 66-67

connector resources, JMS resources and, 66-67

connectors, modules, 165

containers

applet, 95

application client, 95

Enterprise JavaBeans, 95, 98

servlet

See containers, 98

web, 98

containers (*Continued*)

web, 95, 98

CORBA, 163

threads, 165

create-domain command, 34

custom resources, using, 81

D

databases

JNDI names, 79

resource references, 80

supported, 54

delete-domain command, 35

destinations, JMS, overview, 65-66

domains, creating, 34

E

Enterprise Java Beans, threads, 165

Enterprise JavaBeans

activating, 98

authorization, 98

caching, 98

creating, 98

entity, 98

message-driven, 98

passivating, 98

persistent, 98

session, 98

executiontime, 184

external repositories, accessing, 81

F

Foreign Providers, JMS, 69-76

G

get command, monitoring data, 204

H

high availability, 31

HTTP listeners

acceptor threads, 145

default virtual server, 145

overview, 144-146

HTTP service

HTTP listeners, 144-146

Keep-Alive subsystem, 145

request processing threads, 145

virtual servers, 143-144

I

IBM DB2 JDBC driver, 56, 58

IIOP listeners, 164

Inet MSSQL JDBC driver, 60-61

Inet Oracle JDBC driver, 59-60

Inet Sybase JDBC driver, 61

Informix Type 4 JDBC driver, 63

J

Java Business Integration (JBI), See JBI

Environment, 41

Java DB JDBC driver, 55

Java Naming and Directory Service, See JNDI, 98

JavaServer Pages, 98

JCE provider

configuring, 129

JDBC

drivers, 138

supported drivers, 54

JMS

Foreign Providers, 69-76

Resource Adapter, Generic, 69-76

JMS resources

connection factory resources, 65-66

destination resources, 65-66

overview, 65-66

physical destinations, 65-66

queues, 65-66

topics, 65-66

jmsmaxmessagesload, 186

jmsra system resource adapter, 66-67

JNDI, 98

custom resources, using, 81

external repositories, 81

lookups and associated references, 80

names, 79

JSP, See JavaServer Pages, 98

K

Keep-Alive subsystem, HTTP service, 145

kestore.jks file, 115

keypoint intervals, 142

keypoint operations, 142

L

list command, monitoring, 204

list-domains command, 35

log levels, configuring, 172

log records, 169-170

logging

configuring general settings, 172

configuring levels, 172

logger namespaces, 170-171

overview, 169-170

transactions, 141-142

logging (Continued)

- viewing the server log, 173-174

M

- man pages, 27

monitoring

- bean-cache attributes, 186
- container subsystems, 177
- ORB service, 192-193
- transaction service, 193-194
- using get command, 204
- using list command, 204

- MSSQL Inet JDBC driver, 60-61

- MSSQL/SQL Server2000 Data Direct JDBC driver, 57

- MySQL Type 4 JDBC driver, non-XA, 59

N

- naming, JNDI and resource reference, 80

- numbeansinpool, 185

- numexpiredsessionsremoved, 186

- numpassivationerrors, 186

- numpassivations, 186

- numpassivationsuccess, 186

- numthreadswaiting, 185

O

- object request broker, threads, 165

- Object Request Broker (ORB), 163

- overview, 164

- Oracle Data Direct JDBC driver, 56

- Oracle Inet JDBC driver, 59-60

- Oracle OCI JDBC driver, 62-63

- Oracle Thin Type 4 Driver, workaround for, 139

- Oracle Thin Type 4 JDBC driver, 61-62

- oracle-xa-recovery-workaround property, 139

- ORB, 163

- IIOP listeners, 164

- overview, 164

- See object request broker, 165

ORB (Continued)

- service, monitoring, 192-193

P

- performance, thread pools, 165

- Port listeners, 33

Q

- queues, JMS, 65-66

R

- realms, certificate, 112

- request processing threads, HTTP service, 145

- Resource Adapter, Generic, JMS, 69-76

- resource adapters, 138

- jmsra, 66-67

- resource managers, 138

- resource references, 80

- restart server, 36

- rollback

- See transactions

- rolling back, 137

- RSA encryption, 129

S

- server log, viewing, 173-174

- Service Engines, 41

- servlets, 98

- start-domain command, 35

- stop-domain command, 36

- Sybase Data Direct JDBC driver, 57

- Sybase Inet JDBC driver, 61

- Sybase JConnect Type 4 JDBC driver, 58-59

T

- thread pools, 165
 - performance, 165
 - thread starvation, 165
- threads, See thread pools, 165
- topics, JMS, 65-66
- total-beans-created, 186
- totalbeansdestroyed, 186
- totalnumerrors, 184
- totalnumsuccess, 184
- Transaction Manager
 - See transactions
 - managers, 138
- transaction service, monitoring, 193-194
- transactions, 137
 - associating, 138
 - attributes, 138
 - committing, 137
 - completing, 138
 - demarcations, 138
 - distributed, 138
 - logging, 141-142
 - managers, 138
 - recovering, 138, 139-140
 - rolling back, 137
 - timeouts, 140-141
- truststore.jks file, 115

V

- virtual servers, overview, 143-144