Sun Java System Communications Application Server 1.0 High Availability Administration Guide

Beta

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

Part No: 820–4287
March 2008

080619@20490

Contents

Tables

Examples

Preface

This book describes the high-availability features in Application Server, including HTTP load balancing, clusters, session persistence and failover, and High Availability Database (HADB).

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

Communications Application Server Documentation Set

The Communications Communications Application Server documentation set describes deployment planning and system installation. The Uniform Resource Locator (URL) for Communications Application Server documentation is `http://docs.sun.com/coll/1343.8`. For an introduction to Communications Application Server, refer to the books in the order in which they are listed in the following table.

TABLE P–1 Books in the Communications Application Server Documentation Set

Book Title	Description
Documentation Center	Communications Application Server documentation topics organized by task and subject.
Release Notes	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.
Quick Start Guide	How to get started with the Communications Application Server product.
Installation Guide	Installing the software and its components.
Deployment Planning Guide	Evaluating your system needs and enterprise to ensure that you deploy the Communications Application Server in a manner that best suits your site. General issues and concerns that you must be aware of when deploying the server are also discussed.
Application Deployment Guide	Deployment of applications and application components to the Communications Application Server. Includes information about deployment descriptors.

TABLE P–1 Books in the Communications Application Server Documentation Set *(Continued)*

Book Title	Description
Developer's Guide	Creating and implementing Java Platform, Enterprise Edition (Java EE platform) applications intended to run on the Communications Application Server that follow the open Java standards model for Java EE components and APIs. Includes information about developer tools, security, debugging, and creating lifecycle modules.
Java EE 5 Tutorial	Using Java EE 5 platform technologies and APIs to develop Java EE applications.
Java WSIT Tutorial	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.
Administration Guide	System administration for the Communications Application Server, including configuration, monitoring, security, resource management, and web services management.
High Availability Administration Guide	Post-installation configuration and administration instructions for the high-availability database.
Administration Reference	Editing the Communications Application Server configuration file, `domain.xml`.
Upgrade and Migration Guide	Upgrading from an older version of Communications Application Server or migrating Java EE applications from competitive application servers. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
Performance Tuning Guide	Tuning the Communications Application Server to improve performance.
Troubleshooting Guide	Solving Communications Application Server problems.
Error Message Reference	Solving Communications Application Server error messages.
Reference Manual	Utility commands available with the Communications Application Server; written in man page style. Includes the `asadmin` command line interface.

Related Documentation

Communications Application Server can be purchased by itself or as a component of Sun Java Enterprise System (Java ES), a software infrastructure that supports enterprise applications distributed across a network or Internet environment. If you purchased Communications Application Server as a component of Java ES, you should be familiar with the system documentation at `http://docs.sun.com/coll/1286.3`. The URL for all documentation about Java ES and its components is `http://docs.sun.com/prod/entsys.5`.

For documentation about other stand-alone Sun Java System server products, go to the following:

- Message Queue documentation (`http://docs.sun.com/coll/1343.4`)
- Directory Server documentation (`http://docs.sun.com/coll/1224.1`)
- Web Server documentation (`http://docs.sun.com/coll/1308.3`)

A Javadoc™ tool reference for packages provided with the Communications Application Server is located at `http://glassfish.dev.java.net/nonav/javaee5/api/index.html`. Additionally, the following resources might be useful:

- The Java EE 5 Specifications (`http://java.sun.com/javaee/5/javatech.html`)
- The Java EE Blueprints (`http://java.sun.com/reference/blueprints/index.html`)

For information on creating enterprise applications in the NetBeans™ Integrated Development Environment (IDE), see `http://www.netbeans.org/kb/55/index.html`.

For information about the Java DB database included with the Communications Application Server, see `http://developers.sun.com/javadb/`.

The GlassFish Samples project is a collection of sample applications that demonstrate a broad range of Java EE technologies. The GlassFish Samples are bundled with the Java EE Software Development Kit (SDK), and are also available from the GlassFish Samples project page at `https://glassfish-samples.dev.java.net/`.

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

as-install	Represents the base installation directory for Communications Application Server.	Java ES installations on the Solaris™ operating system:
		`/opt/SUNWappserver/appserver`
		Java ES installations on the Linux operating system:
		`/opt/sun/appserver/`
		Other Solaris and Linux installations, non-root user:
		user's-home-directory`/SUNWappserver`
		Other Solaris and Linux installations, root user:
		`/opt/SUNWappserver`
		Windows, all installations:
		SystemDrive`:\Sun\AppServer`

TABLE P–2 Default Paths and File Names *(Continued)*

Placeholder	Description	Default Value

domain-root-dir	Represents the directory containing all domains.	Java ES Solaris installations:
		`/var/opt/SUNWappserver/domains/`
		Java ES Linux installations:
		`/var/opt/sun/appserver/domains/`
		All other installations:
		as-install`/domains/`

domain-dir	Represents the directory for a domain. In configuration files, you might see *domain-dir* represented as follows: `${com.sun.aas.instanceRoot}`	*domain-root-dir*/*domain-dir*
instance-dir	Represents the directory for a server instance.	*domain-dir*/*instance-dir*

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 `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

AaBbCc123	A placeholder to be replaced with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* 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.	`ls [-l]`	The `-l` option is not required.
`{ | }`	Contains a set of choices for a required command option.	`-d {y|n}`	The `-d` option requires that you use either the `y` argument or the `n` argument.
`${ }`	Indicates a variable reference.	`${com.sun.javaRoot}`	References the value of the `com.sun.javaRoot` variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

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/`)

Searching Sun Product Documentation

Besides searching Sun product documentation from the docs.sun.com^{SM} web site, you can use a search engine by typing the following syntax in the search field:

search-term `site:docs.sun.com`

For example, to search for “broker,” type the following:

```
broker site:docs.sun.com
```

To include other Sun web sites in your search (for example, java.sun.com, www.sun.com, and developers.sun.com), use `sun.com` in place of `docs.sun.com` in the search field.

Third-Party Web Site References

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

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

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 820-4287.

CHAPTER 1

High Availability in Application Server

This chapter describes the high availability features in the Sun Java System Communications Application Server that are available with the cluster profile and the enterprise profile.

This chapter contains the following topics.

- "Overview of High Availability" on page 19
- "How Application Server Provides High Availability" on page 22
- "Recovering from Failures" on page 26

Overview of High Availability

High availability applications and services provide their functionality continuously, regardless of hardware and software failures. Such applications are sometimes referred to as providing *five nines* of reliability, because they are intended to be available 99.999% of the time.

Communications Application Server provides the following high availability features:

- High Availability Session Persistence
- High Availability Java Message Service
- RMI-IIOP Load Balancing and Failover

High Availability Session Persistence

Communications Application Server provides high availability of HTTP and SIP requests and session data (both HTTP/SIP session data and stateful session bean data).

Java EE applications typically have significant amounts of session state data. A web shopping cart is the classic example of a session state. Also, an application can cache frequently-needed data in the session object. In fact, almost all applications with significant user interactions need to maintain session state. Both HTTP/SIP sessions and stateful session beans (SFSBs) have session state data.

Preserving session state across server failures can be important to end users. For high availability, Communications Application Server provides the following types of storage for session state data:

- In-memory replication on other servers in the cluster

If the Communications Application Server instance hosting the user session experiences a failure, the session state can be recovered, and the session can continue without loss of information.

For a detailed description of how to set up high availability session persistence, see Chapter 8, "Configuring High Availability Session Persistence and Failover"

High Availability Java Message Service

The Java Message Service (JMS) API is a messaging standard that allows Java EE applications and components to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous. The Sun Java System Message Queue (MQ), which implements JMS, is tightly integrated with Communications Application Server, enabling you to create components that rely on JMS, such as message-driven beans (MDBs).

JMS is made highly available through connection pooling and failover and MQ clustering. For more information, see Chapter 9, "Java Message Service Load Balancing and Failover."

Connection Pooling and Failover

Communications Application Server supports JMS connection pooling and failover. The Communications Application Server pools JMS connections automatically. By default, Communications Application Server selects its primary MQ broker randomly from the specified host list. When failover occurs, MQ transparently transfers the load to another broker and maintains JMS semantics.

For more information about JMS connection pooling and failover, see "Connection Pooling and Failover" on page 161.

MQ Clustering

MQ Enterprise Edition supports multiple interconnected broker instances known as a *broker cluster*. With broker clusters, client connections are distributed across all the brokers in the cluster. Clustering provides horizontal scalability and improves availability.

For more information about MQ clustering, see "Using MQ Clusters with Application Server" on page 162.

RMI-IIOP Load Balancing and Failover

With RMI-IIOP load balancing, IIOP client requests are distributed to different server instances or name servers, which spreads the load evenly across the cluster, providing scalability. IIOP load balancing combined with EJB clustering and availability also provides EJB failover.

When a client performs a JNDI lookup for an object, the Naming Service essentially binds the request to a particular server instance. From then on, all lookup requests made from that client are sent to the same server instance, and thus all `EJBHome` objects will be hosted on the same target server. Any bean references obtained henceforth are also created on the same target host. This effectively provides load balancing, since all clients randomize the list of target servers when performing JNDI lookups. If the target server instance goes down, the lookup or EJB method invocation will failover to another server instance.

IIOP Load balancing and failover happens transparently. No special steps are needed during application deployment. If the Application Server instance on which the application client is deployed participates in a cluster, the Application Server finds all currently active IIOP endpoints in the cluster automatically. However, a client should have at least two endpoints specified for bootstrapping purposes, in case one of the endpoints has failed.

For more information on RMI-IIOP load balancing and failover, see Chapter 10, "RMI-IIOP Load Balancing and Failover."

More Information

For information about planning a high-availability deployment, including assessing hardware requirements, planning network configuration, and selecting a topology, see *Sun Java System Application Server 9.1 Deployment Planning Guide*. This manual also provides a high-level introduction to concepts such as:

- Application server components such as node agents, domains, and clusters
- IIOP load balancing in a cluster
- Message queue failover

For more information about developing applications that take advantage of high availability features, see *Sun Java System Application Server 9.1 Developer's Guide*.

Tuning High Availability Servers and Applications

For information on how to configure and tune applications and Application Server for best performance with high availability, see *Sun Java System Application Server 9.1 Performance Tuning Guide*, which discusses topics such as:

- Tuning persistence frequency and persistence scope
- Checkpointing stateful session beans

- Configuring the JDBC connection pool
- Session size
- Configuring load balancer for best performance

How Application Server Provides High Availability

Application Server provides high availability through the following subcomponents and features:

- "Converged Load Balancer" on page 22
- "HTTP Load Balancer Plug-in" on page 23
- "Storage for Session State Data" on page 24
- "Highly Available Clusters" on page 24

Converged Load Balancer

The converged load balancer accepts both HTTP/HTTPS and SIP/SIPS requests and forwards them to application server instances in a cluster. If an instance fails, becomes unavailable (due to network faults), or becomes unresponsive, the load balancer redirects requests to existing, available machines. The load balancer can also recognize when a failed instance has recovered and redistribute the load accordingly. Application Server provides the converged load balancer. An Application Server instance can be a dedicated load balancer or each instance in a cluster can participate in load balancing, in which case the cluster is *self-load-balancing*.

By distributing workload among multiple physical machines, the load balancer increases overall system throughput. It also provides higher availability through failover of HTTP and SIP requests. For HTTP and SIP session information to persist, you must configure session persistence.

For simple, stateless applications, a load-balanced cluster may be sufficient. However, for mission-critical applications with session state, use load balanced clusters with replicated session persistence.

Server instances and clusters participating in load balancing must have a homogenous environment. Tis means the server instances reference the same server configuration, can access the same physical resources, and have the same applications deployed to them. Homogeneity assures that before and after failures, the load balancer always distributes load evenly across the active instances in the cluster.

For information on configuring load balancing and failover for the converged load balancer, see Chapter 2, "Configuring Converged Load Balancing."

HTTP Load Balancer Plug-in

The HTTP load balancer plug-in accepts HTTP/HTTPS requests and forwards them to application server instances in a cluster. If an instance fails, becomes unavailable (due to network faults), or becomes unresponsive, the load balancer redirects requests to existing, available machines. The load balancer can also recognize when a failed instance has recovered and redistribute the load accordingly. Application Server provides the load balancer plug-in for the Sun Java System Web Server and the Apache Web Server, and Microsoft Internet Information Server.

By distributing workload among multiple physical machines, the load balancer increases overall system throughput. It also provides higher availability through failover of HTTP requests. For HTTP session information to persist, you must configure HTTP session persistence.

For simple, stateless applications a load-balanced cluster may be sufficient. However, for mission-critical applications with session state, use load balanced clusters with replicated persistence.

Server instances and clusters participating in load balancing have a homogenous environment. Usually that means that the server instances reference the same server configuration, can access the same physical resources, and have the same applications deployed to them. Homogeneity assures that before and after failures, the load balancer always distributes load evenly across the active instances in the cluster.

For information on configuring load balancing and failover for the HTTP load balancer plug-in, see Chapter 4, “Configuring HTTP Load Balancing”.

Storage for Session State Data

Storing session state data enables the session state to be recovered after the failover of a server instance in a cluster. Recovering the session state enables the session to continue without loss of information. Application Server provides the following types of high availability storage for HTTP/SIP session and stateful session bean data:

- In-memory replication on other servers in the cluster

In-Memory Replication on Other Servers in the Cluster

In-memory replication on other servers provides lightweight storage of session state data without the need to obtain a separate database. This type of replication uses memory on other servers for high availability storage of HTTP/SIP session and stateful session bean data. Clustered server instances replicate session state in a ring topology. Each backup instance stores the replicated data in memory. Replication of session state data in memory on other servers enables sessions to be distributed.

The use of in-memory replication requires the Group Management Service (GMS) to be enabled. For more information about GMS, see “Group Management Service” on page 103.

If server instances in a cluster are located on different machines, ensure that the following prerequisites are met:

- To ensure that GMS and in-memory replication function correctly, the machines must be on the same subnet.
- To ensure that in-memory replication functions correctly, the system clocks on all machines in the cluster must be synchronized as closely as possible.

Highly Available Clusters

A *cluster* is a collection of Application Server instances that work together as one logical entity. A cluster provides a runtime environment for one or more Java EE applications. A *highly available cluster* integrates a state replication service with clusters and load balancer.

Using clusters provides the following advantages:

- **High availability**, by allowing for failover protection for the server instances in a cluster. If one server instance goes down, other server instances take over the requests that the unavailable server instance was serving.
- **Scalability**, by allowing for the addition of server instances to a cluster, thus increasing the capacity of the system. The load balancer plug-in distributes requests to the available server instances within the cluster. No disruption in service is required as an administrator adds more server instances to a cluster.

All instances in a cluster:

- Reference the same configuration.
- Have the same set of deployed applications (for example, a Java EE application EAR file, a web module WAR or SAR file, or an EJB JAR file).
- Have the same set of resources, resulting in the same JNDI namespace.

Every cluster in the domain has a unique name; furthermore, this name must be unique across all node agent names, server instance names, cluster names, and configuration names. The name must not be `domain`. You perform the same operations on a cluster (for example, deploying applications and creating resources) that you perform on an unclustered server instance.

Clusters and Configurations

A cluster's settings are derived from a named configuration, which can potentially be shared with other clusters. A cluster whose configuration is not shared by other server instances or clusters is said to have a *stand-alone configuration* . By default, the name of this configuration is *cluster_name* `-config`, where *cluster_name* is the name of the cluster.

A cluster that shares its configuration with other clusters or instances is said to have a *shared configuration*.

Clusters, Instances, Sessions, and Load Balancing

Clusters, server instances, load balancers, and sessions are related as follows:

- A server instance is not required to be part of a cluster. However, an instance that is not part of a cluster cannot take advantage of high availability through transfer of session state from one instance to other instances.
- The server instances within a cluster can be hosted on one or multiple machines. You can group server instances across different machines into a cluster.
- A particular load balancer can forward requests to server instances on multiple clusters. You can use this ability of the load balancer to perform an online upgrade without loss of service. For more information, see “Using Multiple Clusters for Online Upgrades Without Loss of Service” in the chapter “Configuring Clusters”
- A single cluster can receive requests from multiple load balancers. If a cluster is served by more than one load balancer, you must configure the cluster in exactly the same way on each load balancer.
- Each session is tied to a particular cluster. Therefore, although you can deploy an application on multiple clusters, session failover will occur only within a single cluster.

The cluster thus acts as a safe boundary for session failover for the server instances within the cluster. You can use the load balancer and upgrade components within the Communications Application Server without loss of service.

Recovering from Failures

Using Sun Cluster

Sun Cluster provides automatic failover of the domain administration server, node agents, Communications Application Serverinstances, and Message Queue. For more information, see *Sun Cluster Data Service for Sun Java System Application Server Guide for Solaris OS*.

Use standard Ethernet interconnect and a subset of Sun Cluster products. This capability is included in Java ES.

Manual Recovery

You can use various techniques to manually recover individual subcomponents:

Remark 1–1 **Should there be a section on recovering the converged load balancer?**

Recovering the Domain Administration Server

Loss of the Domain Administration Server (DAS) affects only administration. Communications Application Server clusters and applications will continue to run as before, even if the DAS is not reachable

Use any of the following methods to recover the DAS:

- Run `asadmin` backup commands periodically, so you have periodic snapshots. After a hardware failure, install App Server on a new machine, with the same network identity and run `asadmin` restore from the back up created earlier. For more information, see "Recreating the Domain Administration Server" on page 28.
- Put the domain installation and configuration on a shared and robust file system (NFS for example). If the primary DAS machine fails, a second machine is brought up with the same IP address and will take over with manual intervention or user supplied automation. Sun cluster uses a similar approach for making DAS fault-tolerant.

- Zip the Communications Application Serverinstallation and domain root directory. Restore it on the new machine, assigning it the same network identity. This may be the simplest approach if you are using the file-based installation.
- Restore from DAS backup. See the AS8.1 UR2 patch 4 instructions

Recovering Node Agents and Server Instances

There are two methods for recovering node agents and sever instances.

Keep a backup zip file. There are no explicit commands to back up the node agent and server instances. Simply create a zip file with the contents of the node agents directory. After failure, unzip the saved backup on a new machine with same host name and IP address. Use the same install directory location, OS, and so on. A file-based install, package-based install, or restored backup image must be present on the machine.

Manual recovery. You must use a new host with the same IP address.

1. Install the Application Server node agent bits on the machine.
2. See the instructions for AS8.1 UR2 patch 4 installation
3. Recreate the node agents. You do not need to create any server instances.
4. Synchronization will copy and update the configuration and data from the DAS.

Recovering the HTTP Load Balancer and Web Server

There are no explicit commands to back up only a web server configuration. Simply zip the web server installation directory. After failure, unzip the saved backup on a new machine with the same network identity. If the new machine has a different IP address, update the DNS server or the routers.

Note – This assumes that the web server is either reinstalled or restored from an image first.

The load balancer plugin (plugins directory) and configurations are in the web server installation directory, typically `/opt/SUNWwbsvr`. The `web-install/web-instance/config` directory contains the `loadbalancer.xml` file.

Recovering Message Queue

Message Queue (MQ) configurations and resources are stored in the DAS and can be synchronized to the instances. Any other data and configuration information is in the MQ directories, typically under `/var/imq`, so backup and restore these directories as required. The new machine must already contain the MQ installation. Be sure to start the MQ brokers as before when you restore a machine.

Using Netbackup

Note – This procedure has not been tested by Sun QA.

Use Veritas Netbackup to save an image of each machine. In the case of BPIP backup the four machines with web servers and Application Servers.

For each restored machine use the same configuration as the original, for example the same host name, IP address, and so on.

For file-based products such as Application Server, backup and restore just the relevant directories. However, for package-based installs such as the web server image, you must backup and restore the entire machine. Packages are installed into the Solaris package database. So, if you only back up the directories and subsequently restore on to a new system, the result will be a "deployed" web server with no knowledge in the package database. This may cause problems with future patching or upgrading.

Do not manually copy and restore the Solaris package database. The other alternative is to backup an image of the machine after the components are installed, for example, web server. Call this the baseline tar file. When you make changes to the web server, back up these directories for example, under `/opt/SUNWwbsvr`. To restore, start with the baseline tar file and then copy over the web server directories that have been modified. Similarly, you can use this procedure for MQ (package-based install for BPIP). If you upgrade or patch the original machine be sure to create a new baseline tar file.

If the machine with the DAS goes down there will be a time when it is unavailable until you restore it.

The DAS is the central repository. When you restore server instances and restart them they will be synchronized with information from the DAS only. Hence, all changes must be performed via `asadmin` or Admin Console.

Recreating the Domain Administration Server

If the machine hosting the domain administration server (DAS) fails, you can recreate the DAS if you have previously backed up the DAS. To recreate a working copy of the 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 Restore the ZIP file onto the third machine.

```
asadmin restore-domain --filename domain-root-dir/sjsas_backup_v00001.zip
--clienthostname machine3 domain1
```

Note – By specifying the `--clienthostname` option, you avoid the need to modify the `jmx-connector` element's `client-hostname` property in the `domain.xml` file.

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 **In the** *domain-root-dir*`/domain1/config/domain.xml` **file on the third machine, update the value of the** `jms-service` **element's** `host` **attribute.**

The original setting of this attribute is as follows:

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

Modify the setting of this attribute as follows:

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

6 **Start the restored domain on machine3:**

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

The DAS contacts all running node agents and provides the node agents with information for contacting the DAS. The node agents use this information to communicate with the DAS.

7 **For any node agents that are not running when the DAS is restarted, change** `agent.das.host` **property value in** *as-install*`/nodeagents/`*nodeagent*`/agent/config/das.properties` **on machine2.**

This step is not required for node agents that are running when the DAS is restarted.

8 **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.

CHAPTER 2

Configuring Converged Load Balancing

This chapter describes the converged load balancer. It includes the following topics:

For information on other types of load balancing, see Chapter 4, "Configuring HTTP Load Balancing," Chapter 9, "Java Message Service Load Balancing and Failover"and Chapter 10, "RMI-IIOP Load Balancing and Failover."

This chapter discusses using the converged load balancer included with the Communications Application Server, which load balances HTTP, HTTPS, SIP, and SIPS requests. Another load balancing option is to use the Sun Secure Application Switch with the Communications Application Server for a hardware-based load balancing solution. For a tutorial on configuring this solution, see the article Clustering and Securing Web Applications: A Tutorial (`http://developers.sun.com/prodtech/appserver/reference/techart/load-balancing.html`).

How the Converged Load Balancer Works

The load balancer attempts to evenly distribute the workload among multiple Application Server instances (either stand-alone or clustered), thereby increasing the overall throughput of the system.

The Converged Load Balancer enables high availability of services deployed on Java EE Application Servers. While doing so, it fails over a session request to another server instance in the same cluster if the original servicing instance is detected to be unavailable or unhealthy to service a request.

Note – The load balancer does not handle URI/URLs that are greater than 8k.

Converged Load Balancer Deployments

You can configure your load balancer in different ways, depending on your goals and environment, as described in the following sections:

- "Using Dedicated Load Balancing Server Instances" on page 32
- "Using Self-Load-Balancing Clusters" on page 32

Using Dedicated Load Balancing Server Instances

In a development environment, you can designate one or more stand-alone server instances as dedicated load balancers, which redirect requests to clusters or to other stand-alone instances that service those requests. These dedicated load balancers are called the *targets* of the load balancer.

The clusters or instances that service requests are called the *LB targets* of the load balancer. The LB targets of a specific load balancer can be clusters or stand-alone instances, but not a mixture of clusters and instances.

- Using Clustered Server Instances as LB Targets — The most common way to deploy the load balancer is with a cluster or clusters of server instances as the LB target or LB targets. By default all the instances in a cluster have the same configuration and the same applications deployed to them. The load balancer distributes the workload between the server instances and requests fail over from an unhealthy instance to a healthy one. If you have multiple clusters, requests can be load balanced across clusters but are only failed over between the instances in a single cluster.
- Using Multiple Stand-Alone Instances as LB Targets — It is also possible to configure your load balancer to use multiple stand-alone instances as LB targets, and to load balance and failover requests between these LB targets. However, in this configuration, you must manually ensure that the stand-alone instances have homogenous environments and the same applications deployed to them. Because clusters automatically maintain a homogenous environment, for most situations it is better and easier to use clusters.

Note – Dedicated load balancers are not supported in a production environment.

Using Self-Load-Balancing Clusters

In a development or production environment, you can designate that all server instances in a cluster participate in both redirecting and servicing requests, without using any dedicated load balancing instances. This is a *self-load-balancing* cluster, in which the target and the LB target are the same cluster.

A front-end hardware IP sprayer distributes request evenly across all instances in the cluster. If you do not use a hardware IP sprayer, a request can be forwarded to any server instance in the cluster. The converged load balancer component on that instance ensures that requests are distributed across the cluster. However, that instance is a single point of failure. The presence of a hardware IP sprayer ensures high availability.

Converged Load Balancing Algorithm

For HTTP and HTTPS requests, the converged load balancer uses a *sticky round robin* algorithm by default. When a new request is sent to the load balancer, it is forwarded to an application server instance based on a simple round robin scheme. If the request is for a session-based application, then this also includes a request for a new session. Subsequent requests from the same client for the same session-based application are considered assigned or sticky requests and are routed by the load balancer to the same instance if that instance is found to be healthy. Hence, the name sticky round robin. Requests to a non-session-based application and the first request for a session-based application are called unassigned requests.

The load balancer automatically uses one of the following to determine HTTP and HTTPS session stickiness:

- Cookies — The load balancer uses a separate cookie to record the route information. The client (typically, the web browser) must support cookies.
- Explicit URL Rewriting — If the client does not support cookies, the sticky information is appended to the URL.

For SIP and SIPS requests, the converged load balancer applies a consistent hashing policy to one or more specified SIP headers that serve as a hash key. The default headers used are `from-tag,to-tag,call-id`.

A DCR file, `data-centric-rules.xml`, provides rules for applying consistent hashing on both HTTP/HTTPS and SIP/SIPS requests. If this file is specified, the instructions in this file override the round-robin and SIP-header-based algorithms. For more information about this file, see "Editing Configuration Settings" on page 36 and "The Data Centric Rules File" on page 40.

Setting Up Converged Load Balancing

This section describes how to set up the load balancer and includes the following sections:

- "Prerequisites for Setting Up Converged Load Balancing" on page 34
- "Procedure to Set Up Converged Load Balancing" on page 34

Prerequisites for Setting Up Converged Load Balancing

Before configuring your load balancer, you must:

- Create Communications Application Server clusters or server instances to participate in load balancing using the cluster profile. For more information, see Chapter 5, "Using Application Server Clusters."
- Create node agents for these clusters or server instances. For more information, see Chapter 7, "Configuring Node Agents."
- Make sure the group management service (GMS) is enabled. It is enabled by default. For more information, see "Group Management Service" on page 103.
- Configure session persistence using in-memory replication. For more information, see Chapter 8, "Configuring High Availability Session Persistence and Failover."
- Deploy applications to these clusters or instances. For more information, see the Application Deployment Guide.

Procedure to Set Up Converged Load Balancing

Use the Admin Console or the `asadmin` command to configure load balancing in your environment. The following sections provide you more information.

▼ To Set Up Converged Load Balancing Using the Admin Console

1 **Create a load balancer.**

On the left frame, click Converged Load Balancers and then click New. In the New Converged Load Balancer page, provide the load balancer name and also select target clusters or instances that will act as load balancers.

Optionally, you can specify the following settings:

- Configuration File Name — Specifies the name of the converged load balancer's configuration file. The default path and name of the configuration file is *domain-dir*/*cluster*/`config/converged-loadbalancer.xml`.
- Automatically Apply Changes — Specifies whether to automatically apply configuration changes to the target server instances. This setting is on by default.
- Self Load Balancing — Specifies whether the target cluster is self-load-balancing. This setting is on by default. For production environments, only self-load-balancing target clusters are supported.

2 **For a load balancer that is not self-load-balancing, add references to clusters or stand-alone server instances for the load balancer to manage.**

On the left frame, click the Converged Load Balancers node and then click the desired load balancer listed under the node. Open the Converged Load Balancer LB Targets tab, click Manage LB Targets and in the Manage LB Targets page, select the required LB targets.

You may select clusters or stand-alone instances as LB targets. Note that you cannot have a combination of both clusters and stand-alone instances as selected LB targets.

Note – This step is not supported in a production environment.

---

3 **If the cluster was already started when you created the load balancer, you must restart the cluster to start the load balancer.**

See Also For more information, see the Admin Console online help.

▼ To Set Up Converged Load Balancing Using the `asadmin` Command

You can perform the following steps using a single `asadmin` command, `create-converged-lb`.

1 **Create a load balancer configuration.**

Use the `asadmin` `create-converged-lb-config` command.

2 **Add a reference to a cluster or stand-alone server instance for the load balancer to manage.**

Use the `asadmin` `create-converged-lb-ref` command.

Example 2–1 Creating a Converged Load Balancer

The following series of `asadmin` commands sets up a cluster, a node agent, and a self-load-balanced converged load balancer.

```
asadmin> create-cluster cluster1
    Command create-cluster executed successfully.
asadmin> create-node-agent nodeagent1
    Command create-node-agent executed successfully.
asadmin> create-node-agent nodeagent2
    Command create-node-agent executed successfully.
asadmin> create-instance --nodeagent nodeagent1 --cluster cluster1 cluster1_instance1
   Command create-instance executed successfully.
asadmin> create-instance --nodeagent nodeagent2 --cluster cluster1 cluster1_instance2
   Command create-instance executed successfully.
asadmin> create-converged-lb --configfile clb.xml --autocommit=true --target cluster1 clb-1
   Command create-converged-lb executed successfully.
asadmin> start-node-agent nodeagent1
```

```
   Command start-node-agent started successfully.
asadmin> start-node-agent nodeagent2
   Command start-node-agent started successfully.
asadmin> start-cluster cluster1
 cluster1_instance1 is running, does not require restart
 cluster1_instance2 is running, does not require restart
 Command start-cluster executed successfully.
```

If the cluster was already started when you created the load balancer, you must restart the cluster to start the load balancer.

See Also For more information about these `asadmin` commands, see the Reference Manual.

Configuring the Converged Load Balancer

These sections describe, in more detail, how to modify and use a load balancer configuration:

- “Editing Load Balancer Settings in the Admin Console” on page 36
- “Editing Load Balancer Settings using the `asadmin set` Command” on page 38
- “Disabling a Server Instance for Load Balancing” on page 39

Editing Load Balancer Settings in the Admin Console

After you have created a converged load balancer, you can edit its settings in the Admin Console as described in the following sections.

- “Editing Configuration Settings” on page 36
- “Editing Load Balancer Details” on page 37
- “Editing Self Load Balancing (LB Target Details)” on page 38

Editing Configuration Settings

To configure a load balancer in the Admin Console, on the left frame, click the Converged Load Balancers node and then click the desired load balancer listed under the node. Open the Settings tab. This displays the Edit Converged Load Balancer Configuration Settings page.

The following table describes the load balancer configuration settings.

TABLE 2–1 Load Balancer Configuration Settings

Setting	Description
Policy Type	Specifies whether the load balancing algorithm is determined by HTTP Policy and SIP Policy or by a DCR file.
HTTP Policy	Specifies the policy to be used for routing HTTP requests. The only allowed value is `round-robin`, which means the load balancer cycles through the cluster's server instances in a specified order.
SIP Policy	Specifies the policy to be used for routing SIP requests. Specifies the parameters on which a consistent hashing policy is applied to obtain the hash key. This can be a single value or comma-separated values of parameter names to hash on. If more than one parameter is specified, the concatenated values of the parameters are used for applying the consistent hashing. The default is `from-tag,to-tag,call-id`.
Upload DCR File	Specifies the DCR file, which stores data centric rules for both HTTP and SIP requests. By default this file is not specified. If this file is specified, the default path and name is *domain-dir*`/`*cluster*`/config/data-centric-rules.xml`. If the converged load balancer configuration that specifies the data centric rules file does not reference any cluster or stand-alone server instance, the default path and name is *domain-dir*`/config/data-centric-rules.xml`. If this file is specified, the instructions in this file override the HTTP Policy and SIP Policy settings. For more information about this file, see "The Data Centric Rules File" on page 40. If an HTTP request doesn't match any DCR file rules, a hash key is generated using the remote host and port. If a SIP request doesn't match any DCR file rules, a hash key is generated using `from-tag,to-tag,call-id`.
Property	Allows you to set property names and values.

Editing Load Balancer Details

You can change converged load balancer settings after you have created the load balancer.

To edit load balancer details in the Admin Console, on the left frame, click the Converged Load Balancers node and then click the desired load balancer listed under the node. Open the Targets tab. Select Edit Load Balancer Details. This displays the Edit Load Balancer Details page.

The following table describes the load balancer details settings.

TABLE 2–2 Load Balancer Details Settings

Setting	Description
Automatically Apply Changes	Specifies whether to automatically apply configuration changes to the target server instances.

TABLE 2–2 Load Balancer Details Settings *(Continued)*

Setting	Description
Configuration File Name	Specifies the name of the converged load balancer's configuration file. The default path and name of the configuration file is *domain-dir*`/`*cluster*`/config/converged-loadbalancer.xml`.
Request Pool Size	Specifies the number of request objects created and pooled by the converged load balancer's proxy.
Send Retries	Specifies the number of retries the proxy attempts with the remote instance when sending of data fails.
Read Timeout	Specifies in milliseconds how long the proxy waits for data from the client in the socket channel.

Editing Self Load Balancing (LB Target Details)

Remark 2–1 **According to the engineers, disabling Self Load Balancing changes the target, not the LB target. This doesn't make sense, because a dedicated load balancer target must be an instance and a self-load-balancing target must be a cluster. Changing the LB target would make more sense, and it's what the Admin Console page title implies is going on, but the configuration would be incomplete unless the user also selected a new target.**

You can change whether an LB target cluster is self-load-balancing after you have created the load balancer.

To edit LB target details in the Admin Console, on the left frame, click the Converged Load Balancers node and then click the desired load balancer listed under the node. Open the LB Targets tab. Select Edit LB Target Details. This displays the Edit LB Target Details page. This page has only one editable setting, Self Load Balancing, which you can enable or disable.

Note – A load balancer with Self Load Balancing disabled is not supported in a production environment.

Editing Load Balancer Settings using the `asadmin set` Command

After you have created a load balancer, you can edit its settings and the settings of its configuration using the `asadmin set` command.

To edit a converged load balancer setting, use the following command:

```
asadmin set config.availability-service.clbname.setting
```

For example:

```
asadmin set config1.availability-service.myclb.config-file=myclb.xml
asadmin set config1.availability-service.myclb.auto-commit=true
asadmin set config1.availability-service.myclb.converged-lb-config-name=myclbcfg
```

To edit a converged load balancer proxy setting, use the following command:

```
asadmin set config.availability-service.clbname.proxy.setting
```

For example:

```
asadmin set config1.availability-service.myclb.proxy.request-pool-size=50
asadmin set config1.availability-service.myclb.proxy.send-retry-count=3
asadmin set config1.availability-service.myclb.proxy.read-time-out=1500
```

To edit a converged load balancer configuration policy setting, use the following command:

```
asadmin set domain.converged-lb-config.clbcfg.converged-lb-policy.setting
```

For example:

```
asadmin set domain1.converged-lb-configs.myclbcfg.converged-lb-policy.http=round-robin
asadmin set domain1.converged-lb-configs.myclbcfg.converged-lb-policy.sip=from-tag,to-tag,call-id
asadmin set domain1.converged-lb-configs.myclbcfg.converged-lb-policy.dcr-file=dcr.xml
```

To edit the `self-loadbalance` setting for a cluster, use the following command:

```
asadmin set domain.converged-lb-config.clbcfg.converged-lb-cluster-ref.cluster.self-loadbalance=setting
```

For example:

```
asadmin set domain1.converged-lb-configs.myclbcfg.converged-lb-cluster-ref.cluster1.self-loadbalance=true
```

Note – A load balancer with Self Load Balancing disabled is not supported in a production environment.

For more information about the `asadmin set` command, see the Reference Manual.

Disabling a Server Instance for Load Balancing

Before you stop a server instance, you should disable the instance for load balancing so that requests are failed over to another instance. To disable a server instance for load balancing, use the `asadmin set` command as follows:

```
asadmin set cluster.server-ref.instance.lb-enabled=false
```

For example:

```
asadmin set cluster1.server-ref.server1.lb-enabled=false
```

For more information about the `asadmin set` command, see the Reference Manual.

Changing the Log Message Level for the Converged Load Balancer

The default log level for converged load balancer messages is set to `INFO`. To change this setting, use the `asadmin set` command to set the `javax.enterprise.system.container.clb` property. For example, change the log level for `config1` to `FINE` as follows:

```
asadmin set
config1.log-service.module-log-levels.property.javax\\.enterprise\\.system\\.container\\.clb=FINE
```

Note that the periods in the property name must be preceded by escape characters. For more information about the `asadmin set` command, see the Reference Manual.

The Data Centric Rules File

The default path and name of the data centric rules file is *domain-dir*`/`*cluster*`/config/data-centric-rules.xml`. If the converged load balancer configuration that specifies the data centric rules file does not reference any cluster or stand-alone server instance, the default path and name is *domain-dir*`/config/data-centric-rules.xml`. The DTD file that validates the file format is *as-install*`/lib/dtds/sun-data-centric-rule_1_0.dtd`.

If a data centric rules file is specified, a consistent hash algorithm determines the server instance to which the request is forwarded. The server instance selection is based on a hash key. The key is extracted from incoming SIP and HTTP requests according to the data centric rules. These rules apply to all applications deployed on the LB targets and can be changed during operation. Only new initial requests are affected by such changes.

The data centric rule language is similar to the triggering language for mapping requests to servlets. The rule language consists of conditions and variables supporting SIP and HTTP key extraction. In contrast to the servlet mapping language, the data centric rules file needs to return a value, either non-null (true) or null (false).

Each incoming SIP and HTTP request is matched with the current rule set. The first rule that matches is used for key extraction. The rules are evaluated sequentially until a condition returns a non-null value. For an OR expression, the first non-null sub-result is returned. For an AND expression, the last sub-result is returned. The key is set to null if no rule matches.

If an HTTP request doesn't match any DCR file rules, a hash key is generated using the remote host and port. If a SIP request doesn't match any DCR file rules, a hash key is generated using `from-tag,to-tag,call-id`.

Here is an example data centric rules file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE user-centric-rules PUBLIC "-//Sun Microsystems Inc.//DTD Sailfin 1.0//EN"
"http://www.sun.com/software/appserver/dtds/sun-data-centric-rule_1_0.dtd">
<user-centric-rules>
  <sip-rules>
    <if>
      <session-case>
        <equal>ORIGINATING</equal>
        <if>
          <header name="P-Asserted-Identity"
              return="request.P-Asserted-Identity.uri.resolve.user">
            <exist/>
          </header>
          <if>
            <header name="P-Asserted-Identity"
                return="request.from.uri.resolve.user">
              <notexist/>
            </header>
            <if>
              <header name="P-Asserted-Identity"
                  return="request.to.uri.resolve.user">
                <notexist/>
              </header>
            </if>
          </if>
        </if>
      </session-case>
      <else return="request.uri.resolve.user" />
    </if>
  </sip-rules>
  <http-rules>
    <or>
      <request-uri return="match.resolve.user">
        <match>/users/([^/]+)</match>
      </request-uri>
      <and>
        <request-uri>
          <match>^/css/</match>
        </request-uri>
        <or>
          <request-uri parameter="referredBy"
              return="parameter.requestUri.uri.resolve.user">
            <exist/>
          </request-uri>
          <request-uri parameter="referredBy"
              return="parameter.from.uri.resolve.user">
            <notexist/>
          </request-uri>
        </or>
      </and>
    </or>
  </http-rules>
</data-centric-rules>
```

The following sections describe the elements in the `data-centric-rules.xml` file.

Top-Level Elements

The top-level elements determine the rules for SIP/SIPS and HTTP/HTTPS requests.

user-centric-rules

This is the top level or root element in the `data-centric-rules.xml` file.

Subelements

You can specify one or both of the following subelements: "sip-rules" on page 42, "http-rules" on page 42.

sip-rules

Determines the data centric rules for SIP and SIPS requests.

Superelements

"user-centric-rules" on page 42

Subelements

All of the following subelements are optional and can occur in any number and any order:

"or" on page 43, "and" on page 43, "if" on page 44, "header" on page 44, "request-uri" on page 45, "session-case" on page 45, "cookie" on page 46

http-rules

Determines the data centric rules for HTTP and HTTPS requests.

Superelements

"user-centric-rules" on page 42

Subelements

All of the following subelements are optional and can occur in any number and any order:

"or" on page 43, "and" on page 43, "if" on page 44, "header" on page 44, "request-uri" on page 45, "session-case" on page 45, "cookie" on page 46

Operator Elements

The operator elements specify decisions between different rules. The `or`, `and`, and `if` elements are recursive. You can specify any number of branches and levels of these elements, as long as the lowest level of each branch contains one of the "Condition Elements" on page 44.

or

Evaluates to true if and only if at least one contained condition evaluates to a non-null value.

Superelements

"sip-rules" on page 42, "http-rules" on page 42, "or" on page 43, "and" on page 43, "if" on page 44

Subelements

All of the following subelements are optional and can occur in any number and any order:

"or" on page 43, "and" on page 43, "if" on page 44, "header" on page 44, "request-uri" on page 45, "session-case" on page 45, "cookie" on page 46

and

Evaluates to true if and only if all contained conditions evaluate to a non-null value.

Superelements

"sip-rules" on page 42, "http-rules" on page 42, "or" on page 43, "and" on page 43, "if" on page 44

Subelements

All of the following subelements are optional and can occur in any number and any order:

"or" on page 43, "and" on page 43, "if" on page 44, "header" on page 44, "request-uri" on page 45, "session-case" on page 45, "cookie" on page 46

if

Contains a single condition. If the condition evaluates to a non-null value, the evaluation continues in the `if` branch. If the condition evaluates to null, the evaluation continues in the optional `else` branch.

Superelements

"sip-rules" on page 42, "http-rules" on page 42, "or" on page 43, "and" on page 43, "if" on page 44

Subelements

All of the following subelements are optional and can occur in any number and any order:

"or" on page 43, "and" on page 43, "if" on page 44, "header" on page 44, "request-uri" on page 45, "session-case" on page 45, "cookie" on page 46

The "else" on page 44 subelement is optional and must occur last if specified.

else

Specifies an alternative when the parent `if` element's condition evaluates to null.

Superelements

"if" on page 44

Attributes

The following attribute is required and case-sensitive.

`return`	Specifies a variable that evaluates to the return value. See "Variables" on page 48.

Condition Elements

The condition elements specify the kind of data on which rule decisions are based. All attribute values are case-sensitive.

header

Specifies a rule based on a request header.

Superelements

"sip-rules" on page 42, "http-rules" on page 42, "or" on page 43, "and" on page 43, "if" on page 44

Subelements

Exactly one of the following subelements is required:

"exist" on page 47, "notexist" on page 47

Attributes

The following attributes are required.

`name`	Specifies the name of the request header.
`return`	Specifies a variable that evaluates to the return value. See "Variables" on page 48.

request-uri

Specifies a rule based on a request URI and its parameters.

Superelements

"sip-rules" on page 42, "http-rules" on page 42, "or" on page 43, "and" on page 43, "if" on page 44

Subelements

Exactly one of the following subelements is required:

"exist" on page 47, "notexist" on page 47, "match" on page 47

Attributes

The following attributes are required if the subelement is `exist` or `notexist` and optional if the subelement is `match`.

`parameter`	Specifies the request URI parameter.
`return`	Specifies a variable that evaluates to the return value. See "Variables" on page 48.

session-case

Specifies a rule based on the `call` parameter of a SIP session. Only relevant in IMS/3GPP (Internet Protocol Multimedia Subsystem Third Generation Partnership Project) environments. For details, see the "equal" on page 47 subelement description.

Superelements

"sip-rules" on page 42, "http-rules" on page 42, "or" on page 43, "and" on page 43, "if" on page 44

Subelements

The "equal" on page 47 subelement is required and must occur first.

All of the following subelements are optional and can occur in any number and any order:

"or" on page 43, "and" on page 43, "if" on page 44, "header" on page 44, "request-uri" on page 45, "session-case" on page 45, "cookie" on page 46

cookie

Specifies a rule based on an HTTP cookie.

Superelements

"sip-rules" on page 42, "http-rules" on page 42, "or" on page 43, "and" on page 43, "if" on page 44

Subelements

Exactly one of the following subelements is required:.

"exist" on page 47, "notexist" on page 47

Attributes

The following attributes are required.

`name`	Specifies the name of the cookie.
`return`	Specifies a variable that evaluates to the return value. See "Variables" on page 48.

Condition Type Elements

The condition type elements compare request data to the data expected by the rules. All comparisons are case-sensitive.

exist

Specifies that a variable must evaluate to a `header`, `request-uri`, or `cookie` that exists in the request. See “Variables” on page 48.

Superelements

“header” on page 44, “request-uri” on page 45, “cookie” on page 46

notexist

Specifies that a variable must evaluate to a `header`, `request-uri`, or `cookie`that does not exist in the request. See “Variables” on page 48.

Superelements

“header” on page 44, “request-uri” on page 45, “cookie” on page 46

equal

Specifies which part of a call is processed, the originating or terminating side of the call. Only relevant in IMS/3GPP environments. Allowed values are:

- `EXTERNAL` — Specifies that there is no route header in the SIP request or no `call` parameter in the route header.
- `ORIGINATING` — Specifies that the route header contains `call=orig`.
- `TERMINATING` — Specifies that the route header contains `call=term_registered`.
- `TERMINATING_UNREGISTERED` — Specifies that the route header contains `call=term_unregistered`.

Superelements

“session-case” on page 45

match

Specifies the regular expression that a `request-uri` must match. The return value is the match of the first capturing group in the regular expression. For more information about regular expressions, see
`http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html`.

The match element may have an optional prefix that is prepended to the return value. **[Remark 2–2 : What are the allowed prefix values and what is their syntax?]**

Superelements

"request-uri" on page 45

Variables

The name, parameter, and return attributes and the "exist" on page 47 and "notexist" on page 47 elements in the data centric rules file use variables. Variables containing the string resolve contain values retrieved by performing ENUM lookups of TEL URIs. Some variables contain *replaceable* text. For example, in the request.*header* variable, the *header* is replaced by the name of a SIP or HTTP header. The syntax for matching SIP and HTTP requests is slightly different.

The following SIP variables are supported:

```
request.uri
request.uri.scheme
request.uri.user
request.uri.host
request.uri.port
request.method

request.uri.resolve
request.uri.resolve.user
request.uri.resolve.host

request.header
request.header.uri
request.header.uri.scheme
request.header.uri.user
request.header.uri.host
request.header.uri.port
request.header.uri.display-name

request.header.uri.resolve
request.header.uri.resolve.user
request.header.uri.resolve.host

request.header.match
request.header.match.resolve.user

match
match.resolve.user
```

The following HTTP variables are supported:

```
request.header
request.header.uri
request.header.uri.user
request.header.uri.host
request.header.uri.resolve
request.header.uri.resolve.user
request.header.uri.resolve.host

parameter.parameter
parameter.parameter.uri
parameter.parameter.uri.user
parameter.parameter.uri.host
parameter.parameter.uri.resolve
parameter.parameter.uri.resolve.user
parameter.parameter.uri.resolve.host

match
match.resolve.user

cookie.cookie-name
```

The resolution of the HTTP variable `parameter.`*parameter*`.uri.resolve.user` is complex. The variable matches a parameter value in an HTTP request, and this value may be a single `name-addr` or a comma-separated sequence of them. The `name-addr` elements are resolved until a usable user centric hash key is found. The order of resolution is as follows:

1. If a `name-addr` contains a `user=phone` parameter, it is resolved as a TEL URL, otherwise the user part of the URI is extracted. Resolution of a SIP URI may thus fail if it specifies a telephone number entity that cannot be resolved by ENUM, or else because there is no user part present in the SIP URI.
2. If all SIP URIs have been considered, a second attempt is made, and the TEL URLs are read from left to right. Evaluation stops as soon as a usable user centric key has been found.
3. If every resolution attempt fails, no user centric key is found. If an HTTP request doesn't match any DCR file rules, a hash key is generated using the remote host and port. If a SIP request doesn't match any DCR file rules, a hash key is generated using `from-tag,to-tag,call-id`.

For example, if the variable is `parameter.from.uri.resolve.user` and the HTTP request is `GET ...?...&from=...&...HTTP/1.1`, the outcome may be according to the values in the following table. Some of the characters in the examples may in reality need to be URL-encoded (< would appear as `%3C` and so on.)

TABLE 2–3 Examples of `from` Parameter Values

Value of `from` Parameter	User Centric Key
`<sip:server.xx.yy>`	none
`<sip:alice@server.xx.yy>`	`alice`
`<tel:+1-333-555>,<sip:+1-22-22@server.xx.yy;user=phone>`	from ENUM

◆ ◆ ◆ CHAPTER 3

Configuring Web Servers for HTTP Load Balancing

This chapter explains how to configure the web servers supported by the HTTP load balancer plug-in available with Application Server 9.1.1 and GlassFish V2. The HTTP load balancer plug-in available with Application Server 9.1.1 supports the following web servers:

- Sun Java System Web Server 6.1 and 7.0
- Apache Web Server 2.0.x
- Microsoft IIS 5.0 and 6.0

Note – The converged load balancer does not use a web server. For more information about the converged load balancer, see Chapter 2, "Configuring Converged Load Balancing."

Note – GlassFish V2 supports only Sun Java System Web Server (versions 6.1 and 7.0). To use the HTTP load balancer plug-in with GlassFish V2, you need to manually install and configure the HTTP load balancer plug-in. For more information about installing the HTTP load balancer plug-in with GlassFish V2, see Chapter 1, "Installing Application Server Software," in *Sun Java System Application Server 9.1 Installation Guide*.

The HTTP load balancer plug-in installation program, which is a part of the Application Server 9.1 installation program, makes a few modifications to the web server's configuration files. These changes depend upon the web server you are using. In addition, for some web servers you must make manual configurations in order for the HTTP load balancer to work properly.

Note – The HTTP load balancer plug-in can be installed either along with Sun Java System Communications Application Server 1.0, or separately, on a machine running the supported web server. For complete details on the installation procedure, see Chapter 1, "Installing Application Server Software," in *Sun Java System Application Server 9.1 Installation Guide*.

- "Configuring Sun Java System Web Server" on page 52

- "Using Apache Web Server" on page 59
- "Using Microsoft IIS" on page 68

Configuring Sun Java System Web Server

For Sun Java System Web Server, when you install the load balancer plug-in using the Sun Java System Application Server 9.1 installation wizard, the installation wizard automatically does all the necessary configuration. No manual configuration is required. The load balancer plug-in bundled with Application Server 9.1 supports the following versions of Sun Java System Web Server:

- Sun Java System Web Server 6.1
- Sun Java System Web Server 7.0

But, if you are using GlassFish V2, you must download the Application Server load balancer plug-in separately from `http://download.java.net/javaee5/external/SunOS_X86/aslb/jars/aslb-9.1-MS4-b7.jar` and make some manual changes to set it up. For detailed steps on how to install and set up the plug-in for GlassFish V2, refer to the *Sun Java System Application Server 9.1 Installation Guide*.

▼ To Configure Sun Java System Web Server

Before You Begin

Note – The following steps are automatically performed by the installation program for Application Server 9.1. But, if you are using Glassfish V2, you will need to perform these steps manually.

1 To the web server instance's `magnus.conf` file, add the following lines:

```
##BEGIN EE LB Plug-in Parameters
Init fn="load-modules"
shlib="web-server-install-dir/plugins/lbplugin/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrough" Thread="no"
Init fn="init-passthrough"
##END EE LB Plug-in Parameters=
```

2 Append the following line if it does not exist already:

```
Init fn="load-modules" shlib=".../libj2eeplugin.so" shlib_flags="(global|now)"
```

3 In the file `web-server-install-dir/config/obj.conf`, insert the following in a single line before the first occurrence of the string `nametrans`:

```
Nametrans fn="name-trans-passthrough" name="lbplugin"
config-file="web-server-install-dir/config/loadbalancer.xml"
```

The order in which `NameTrans` entries appear in `obj.conf` is very important. The installer puts the `NameTrans` entries in the correct location, but if you are editing `obj.conf` for other purposes you must ensure that the order remains correct. In particular, the load balancer info must come before the `document-root` function. For more information on the `obj.conf` file, see *Sun Java System Web Server 7.0 Administrator's Configuration File Reference*.

4 **Append the following lines to the file** `web-server-install-dir/config/obj.conf`**:**

```
<Object name = "lbplugin">
ObjectType fn="force-type" type="magnus-internal/lbplugin"
PathCheck fn="deny-existence" path="*/WEB-INF/*"
Service type="magnus-internal/lbplugin" fn="service-passthrough"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</Object>
```

5 **Edit the** *web-server-install-dir*`/start` **script to update the** `LD_LIBRARY_PATH` **value to include** *app-server-install-dir*`/lib/lbplugin/lib`**.**

The *app-server-install-dir*`/lib/lbplugin/lib` directory contains binaries that the load balancer plug-in requires.

6 **(Optional) For the new DAS-based Load Balancer Administration, configure the web server for SSL.**

For detailed instructions for Web Server 6.1 , see "To Set Up the HTTP Load Balancer in SSL Mode for Sun Java System Web Server 6.1" on page 54.

For detailed instructions for Web Server 7, see "To Set up the HTTP Load Balancer in SSL Mode for Sun Java System Web Server 7 " on page 57.

7 **If the web server is not already running, start the web server.**

Configuring Sun Java System Web Server to Use Auto Apply

Auto Apply is a feature provided by Application Server 9.1 to send the load balancer configuration automatically over the wire to the web server configuration directory. The following procedures explain how to configure Sun Java System Web Server (versions 6 and 7) to use this feature.

▼ To Set Up the HTTP Load Balancer in SSL Mode for Sun Java System Web Server 6.1

Note – You need to perform the steps in this section only if you want to use the Auto Apply feature of the load balancer plug-in. This feature helps to send the load balancer configuration automatically over the wire to the web server configuration directory.

---

1. **Using a browser, access the Admin GUI of Web Server and login.**

2. **Select your server instance and click on Manage.**

3. **Click on the Security tab.**

4. **Initialize the trust database by giving the username and password. This could be done using either the `certutil` command or the GUI. The following options of the `certutil` command could be used to initialize the trust database:**

   ```
   certutil -N -P  "https-instance-name-hostname-" -d .
   ```

   - When prompted by certutil, enter the password to encrypt your keys. Enter a password, which will be used to encrypt your keys. The password should be at least eight characters long, and should contain at least one non-alphabetic character.
   - When prompted to enter a new password, specify your password.

5. **Create a sample local Certificate Authority (CA) using the following command:**

   ```
   certutil -S -P "https-boqueron.virkki.com-boqueron-"
   -d . -n SelfCA -s "CN=Self CA,OU=virkki.com,C=US"
   -x -t "CT,CT,CT"
   -m 101 -v 99 -5
   ```

   a. **When prompted to enter 0-7 for the type of certificate, type 5 for SSL CA. When the prompt reappears, specify 9.**

   b. **When queried "Is this a critical extension [y/n]?," specify "y."**

6. **Use the above sample CA to generate a certificate**

   ```
   certutil -S -P "https-instance-name-hostname-"
   -d . -n MyServerCert -s "CN=boqueron.virkki.com,C=US"
   -c SelfCA -t "u,u,u" -m 102 -v 99 -5
   ```

   a. **When prompted to enter 0-7 for the type of certificate, type 1 for SSL Server. When the prompt reappears, specify 9.**

   b. **When queried "Is this a critical extension [y/n]?," specify "y."**

7 **Edit the current HTTP Listener socket by clicking on Preferences→Edit Listen Socket. Enable the security and choose the certificate created in the previous step.**

[Remark 3–2 : Need more info about where/which file the following edits are done.] If you wish to not use the GUI, change the entry to read as follows : Change the tag so that the value of security is "true." The tag must be altered to contain additional body content and a closing tag. Be sure to remove carriage returns when adding the tag.

```
LS id="ls1" port="80" servername="$DEPLOY-INSTANCE"
defaultvs="https-$DEPLOY-INSTANCE" ip="any" security="true"
acceptorthreads="1" blocking="false">
<SSLPARAMS servercertnickname="$HOST-DOMAIN" ssl2="off"
ssl2ciphers="-rc4,-rc 4export,-rc2,-rc2export,-desede3,-des"
ssl3="on"
tls="on"
ssl3tlsciphers="-rsa_rc4_128_sha,+rsa_rc4_128_md5,-rsa_rc4_56_sha,-rsa_rc4_40_md5,
+rsa_3des_sha,+rsa_des_sha,-rsa_des_56_sha,-rsa_rc2_40_md5,
-rsa_null_md5,-fortezza,-fortezza_rc4_128_sha,-fortezza_null,
+fips_3des_sha,-fips_des_sha" tlsrollback="on" clientauth="off"/>
</LS>
```

▼ To Export and Import the DAS Certificate for Sun Java System Web Server 6.1

1 **If you are using Application Server 9.1, export the DAS certificate by executing the command:**

```
<as home>/lib/upgrade/pk12util -d <domain root>/config -o sjsas.p12-W
<file password> -K <master password> -n s1as
```

- **If you are using GlassFish V2, you must use the following commands to export the DAS certificate:**

  ```
  <JAVA_HOME>/bin/keytool -export -rfc -alias s1as -keystore
  <GLASSFISH_HOME>/domains/<DOMAIN_NAME>/config/keystore.jks-file s1as.rfc
  ```

  where, <GLASSFISH_HOME> indicates the Application Server installation directory and <DOMAIN_NAME> refers to the domain whose certificate is being exported.

- **Copy the certificate file to the web server configuration directory.**

2 **If you are using Application Server 9.1, import the DAS certificate into the Web Server instance using the following commands:**

```
<webserver home>/bin/https/admin/bin/pk12util-i sjsas.p12-d <webserver
home>/alias -W<file password> -K <webserver security db password> -P
<instance-name>-<hostname>-

<webserver home>/bin/https/admin/bin/certutil -M -n s1as -t "TCu,Cu,Tuw"
-d alias -P <instance-name>-<hostname>-
```

This command makes the Application Server CA be a trusted CA to sign both client and server certificates.

- **If you are using GlassFish V2, import the DAS certificate from the `rfc` file created using certutil, the NSS security tool.**

```
<webserver_home>/bin/certutil -A -a -n s1as -t "TCu,Cu,Tuw" -i s1as.rfc -d alias -P <instance-name>-<hostname>-
```

  where, <webserver_home> refers to the web server installation directory.

  You can check the presence of this certificate by using the following command, which would list the s1as certificate along with other CA certificates including the default server certificate. Ensure that you type the command in a single line.

```
<WS_INSTALL_ROOT>/bin/certutil -L -d
<WS_INSTALL_ROOT>/admin-server/config-store/
<DEFAULT_CONFIG_NAME>/config
```

3 **If `obj.conf` does not contain the following lines, please append them at the end of the file. If you are using Application Server 9.1, this step is automatically performed by the installation program.**

```
<Object ppath="*lbconfigupdate*">
PathCheck fn="get-client-cert" dorequest="1" require="1"
<Object>
<Object ppath="*lbgetmonitordata*">
PathCheck fn="get-client-cert" dorequest="1" require="1"
</Object>
```

4 **You can verify the above set up from the DAS using the steps provided in the section "Verifying the Setup" on page 67. Instead of using the local CA, you can use any other CA and server certificate. In that case you can skip steps 5 and 6 listed in the previous section, but need to import the server certificate that you obtained from other CAs.**

▼ To Set up the HTTP Load Balancer in SSL Mode for Sun Java System Web Server 7

1 **(Optional) Create the NSS database using the following command. This step is not needed if the NSS database exists. Make sure that you type the command in a single line.**

```
webserver-install-dir/bin/certutil -N -d
/webserver-install-dir/admin-server/config-store/config-name/config
```

When prompted, provide the NSS database password.

2 **Start the Admin Server using the following command.**

```
webserver-install-dir/admin-server/bin/startserv.bat
```

3 **Create a self-signed certificate using the following command. Make sure that you type the command in a single line.**

```
webserver-install-dir/bin/wadm create-selfsigned-cert --user=
admin-user --server-name=host-name
--nickname=ServerCert --token=internal --config=config-name
```

4 **Create an HTTP listener using the following command. Make sure that you type the command in a single line.**

```
webserver-install-dir/bin/wadm create-http-listener
--user=admin-user --server-name=host-name
--default-virtual-server-name=default-virtual-server-name
--listener-port=8090 --config=config-name http-listener-ssl
```

5 **Enable the SSL and assign certificate using the following command. Make sure that you type the command in a single line.**

```
webserver-install-dir/bin/wadm set-ssl-prop
--user=admin-user --http-listener=http-listener-ssl
--config=config-name enabled=true server-cert-nickname=ServerCert
```

▼ To Export and Import the DAS Certificate

1 **Export the DAS certificate.**

- **If you are using Application Server 9.1, export the DAS certificate by executing the command:**

  ```
  <as home>/lib/upgrade/pk12util -d <domain root>/config -o sjsa.p12 -W
  <file password> -K <master password> -n s1as
  ```

- **If you are using GlassFish V2, export the DAS certificate, named with the alias `s1as` using the Java SE 5.0 security tool called `keytool`. While doing so, select the `-rfc` option to export the certificate in the printable encoding format, as defined by the Internet RFC 1421 standard.**

  From the command line, you can use the following commands to export the DAS certificate:

  ```
  <JAVA_HOME>/bin/keytool -export -rfc -alias s1as -keystore
  <GLASSFISH_HOME>/domains/<DOMAIN_NAME>/config/keystore.jks-file s1as.rfc
  ```

  where, <GLASSFISH_HOME> indicates the Application Server installation directory and <DOMAIN_NAME> refers to the domain whose certificate is being exported.

Copy the certificate file to the web server configuration directory.

2 **Import the DAS certificate into the Web Server.**

- **If you are using Application Server 9.1, import the DAS certificate into the Web Server instance using the following commands:**

  ```
  <webserver home>/bin/https/admin/bin/pk12util-i sjsas.p12 -d <webserver
  home>/alias -W<file password> -K <webserver security db password> -P
  <instance-name>-<hostname>-

  <webserver home>/bin/https/admin/bin/certutil -M -n s1as -t "TCu,Cu,Tuw"
  -d alias -P <instance-name>-<hostname>-
  ```

  This command makes the Application Server CA be a trusted CA to sign both client and server certificates.

- **If you are using GlassFish V2, import the DAS certificate from the `rfc` file created using `certutil`, the NSS security tool.**

  ```
  <webserver_home>/bin/certutil -A -a -n s1as -t "TC" -i s1as.rfc -d
  <WS_INSTALL_ROOT>/admin-server/config-store/<CONFIG_NAME>/config
  ```

  where, <webserver_home> refers to the web server installation directory and <CONFIG_NAME> refers to the configuration name created for the default web server instance.

  You can check the presence of this certificate by using the following command, which would list the `s1as` certificate along with other CA certificates including the default server certificate. Make sure that you type the entire command in a single line.

  ```
  <WS_INSTALL_ROOT>/bin/certutil -L -d
  <WS_INSTALL_ROOT>/admin-server/config-store/
  <DEFAULT_CONFIG_NAME>/config
  ```

  You can also use the Web Server Admin Console to view this. Select the configuration to which the certificate has been imported to (default config, in this case), and then select the Certificates tab. To look at all the certificates available, select the Certificate Authorities sub tab.

3 **Make the following configuration changes to Web Server 7.0.**

a. **Append the following lines to `obj.conf`file located at** `<WS_INSTALL_ROOT>/admin-server/config-store/<DEFAULT_CONFIG_NAME>/config/:`

```
 <Object ppath="*lbconfigupdate*">
 PathCheck fn="get-client-cert" dorequest="1" require="1"
</Object>
<Object ppath="*lbgetmonitordata*">
 PathCheck fn="get-client-cert" dorequest="1" require="1"
</Object>
```

4 **Deploy the configuration. While doing the changes listed in the previous steps, the Admin Console would mark this configuration to be deployed.**

a. **Select the icon for Deployment Pending in the Web Server Admin Console. You can also deploy this configuration using the CLI utility `wadm` as follows:**

```
<WS_INSTALL_ROOT>/bin/wadm deploy-config-user=<admin><DEFAULT_CONFIG_NAME>
```

where <admin> is the administrator user name.

5 **Test this setup from the GlassFish DAS to see if it communicates with the configured HTTP Load Balancer over SSL. For more information, see "Verifying the Setup" on page 67.**

Using Apache Web Server

The load balancer plug-in bundled with Application Server 9.1 supports Apache Web Server 2.0.x. To use Apache Web Server, you must perform certain configuration steps before and after installing the load balancer plug-in. The load balancer plug-in installation also makes additional modifications to the Apache Web Server. After the plug-in is installed, you must perform additional configuration steps.

Note – Apache 2 has multithreaded behavior if compiled with the `--with-mpm=worker` option.

- "Requirements for Using Apache Web Server" on page 60
- "Configuring Apache before Installing the HTTP Load Balancer Plug-in" on page 61
- "Modifications Made by the HTTP Load Balancer Plug-in Installer" on page 65
- "Configuring Apache After Installing the HTTP Load Balancer Plug-In" on page 65
- "Starting Apache on Solaris and Linux" on page 67

Requirements for Using Apache Web Server

For the Apache Web Server, your installation must meet the minimum requirements.

With Apache, the load balancer plug-in requires:

- `openssl-0.9.7e (source)`
- `httpd-2.0.59 (source)`
- `gcc-3.3-sol9-sparc-local` packages (for Solaris 9 SPARC).
- `gcc-3.3-sol9-intel-local` packages (for Solaris 9 x86)
- The pre-installed `gcc` (for Solaris 10)
- `flex-2.5.4a-sol9-sparc-local` packages (for Solaris 9 SPARC)
- `flex-2.5.4a-sol9-intel-local` packages (for Solaris 9 x86)
- The pre-installed `flex` (for Solaris 10)

The software sources are available at `http://www.sunfreeware.com`

In addition, before compiling Apache:

- On the Linux platform, install Sun Java System Communications Application Server on the same machine.
- On the Solaris 9 operating system, use `pkgadd` to install `gcc` and `flex.` Note that `pkgadd` requires root access.
- On the Solaris 9 operating system, ensure that `gcc` version 3.3 and `make` are in the `PATH`, and `flex` is installed.
- On the Solaris 10 operating system, before running `make` for OpenSSL, run `mkheaders`, located in `/usr/local/lib/gcc-lib/sparc-sun-solaris2.9/3.3/install-tools` on Solaris SPARC or `/usr/local/lib/gcc-lib/i386-pc-solaris2.9/3.3/install-tools` on Solaris x86.
- If you are using `gcc` on Red Hat Enterprise Linux Advanced Server 2.1, the version must be later than `gcc` 3.0.

Note – To use a C compiler other than `gcc`, set the path of the C compiler and `make` utility in the PATH environment variable.

Applying the Apache Web Server Patch

Before installing the load balancer plug-in for Apache, apply the patch for the Apache Web Server issue 12355. More details about this issue are available at `http://issues.apache.org/bugzilla/show_bug.cgi?id=12355`. This patch is required for the Auto Apply feature to work. To apply the patch, follow these steps.

1. Untar `http-2.0.59.tar` and go to the directory httpd-2.0.59.

2. Download the patch from `http://issues.apache.org/bugzilla/attachment.cgi?id=16495` and save it as a file, for example, `12355.diff`.
3. From the directory `httpd-2.0.59/modules/ssl`, run the following command:

```
patch < 12355.diff
```

Configuring Apache before Installing the HTTP Load Balancer Plug-in

The Apache source must be compiled and built to run with SSL. This section describes the minimum requirements and high-level steps needed to successfully compile Apache Web Server to run the load balancer plug-in. These requirements and steps only apply to the Solaris and Linux versions of the software. For information on the Windows version of Apache, see the Apache web site.

Note – The instructions included here are adapted from the instructions at `http://httpd.apache.org/docs`. For detailed instructions on installing SSL-aware Apache, please see that web site.

▼ To Install SSL-aware Apache

Before You Begin You must have already downloaded and uncompressed the Apache software.

1 **Download and unpack the OpenSSL source, available at `http://openssl.org`.**

2 **Compile and build OpenSSL.**

For full installation instructions, see the file named `INSTALL` in the directory where you uncompressed OpenSSL. That file has information on installing OpenSSL in a user-specified location.

For more information about OpenSSL, see the `http://www.openssl.org/`.

3 **Download and unpack Apache.**

Apache is available from `http://httpd.apache.org`.

4 **Compile and build Apache. Configure the source tree:**

a. **`cd http-2.0_`*x*.**

b. **Run the following command:**

`./configure --with-ssl=` *OpenSSL-install-path* `--prefix=` *Apache-install-path* `--enable-ssl --enable-so`

In the above commands, *x* is the Apache version number, *open-ssl-install-path* is the absolute path to directory where OpenSSL is installed, and *Apache-install-path* is the directory in which to install Apache.

Note that you only need to use the `--enable-ssl --enable-so` options if your Apache 2 server will be accepting HTTPS requests.

5 **For Apache on Linux 2.1, before compiling:**

a. **Open** `src/MakeFile` **and find the end of the automatically generated section.**

b. **Add the following lines after the first four lines after the automatically generated section:**

```
LIBS+= -licuuc -licui18n -lnspr4 -lpthread -lxerces-c
-lsupport -lnsprwrap -lns-httpd40
LDFLAGS+= -L/application-server-install-dir/lib -L/opt/sun/private/lib
```

Note that `-L/opt/sun/private/lib` is only required if you installed Application Server as part of a Java Enterprise System installation.

For example:

```
## (End of automatically generated section)
##
CFLAGS=$(OPTIM) $(CFLAGS1) $(EXTRA_CFLAGS)
LIBS=$(EXTRA_LIBS) $(LIBS1)
INCLUDES=$(INCLUDES1) $(INCLUDES0) $(EXTRA_INCLUDES)
LDFLAGS=$(LDFLAGS1) $(EXTRA_LDFLAGS)
"LIBS+= -licuuc -licui18n -lnspr4 -lpthread
-lxerces-c -lsupport -lnsprwrap -lns-httpd40
LDFLAGS+= -L/application-server-install-dir /lib -L/opt/sun/private/lib
```

c. **Set environment variable** `LD_LIBRARY_PATH`**.**

With stand–alone installations, set it to the Communications Application Server: *as-install*`/lib`

With Java Enterprise System installations, set it to the Communications Application Server: *as-install*`/lib:opt/sun/private/lib`.

If you are using Solaris 9, add `/usr/local/lib` to the `LD_LIBRARY_PATH`.

6 **Compile Apache as described in the installation instructions for the version you are using.**

For more information, see the `http://httpd.apache.org/`

In general, the steps are:

a. `make`

b. `make install`

7 **Make sure Apache's `ssl.conf` and `httpd.conf` files contain the correct values for your environment.**

- In `ssl.conf`, for `VirtualHost default:`*port* replace the default hostname and port with the hostname of the local system where Apache is installed and the server's port number.

  Without this change, the load balancer will not work. On Solaris Apache may not start and on Linux, HTTPS requests may not work.
- In `ssl.conf`, for `ServerName www.example.com:443`, replace `www.example.com` with the hostname of the local system where Apache is installed.

  Without this change, the following warning appears when you start Apache if a security certificate is installed:

  ```
  [warn] RSA server certificate CommonName (CN)
  hostname does NOT match server name!
  ```

  For more information on installing certificates for Apache, see "To Create a Security Certificate for Apache " on page 65.
- In `httpd.conf`, for `ServerName www.example.com:80`, replace `www.example.com` with the hostname of the local system where Apache is installed.

  Without this change, you see warnings when you start Apache that the system could not determine the server's fully qualified domain name, and that there are overlapping VirtualHost entries.

8 **Ensure that the Apache user has the required access permissions to the *apache-install-location*`/conf/` directory and files in this directory.**

The Apache user is the UNIX user under which the Apache server responds to requests. This user is defined in the file `httpd.conf`.

If you installed Apache as a root user, read the note about configuring the Apache user and group in *apache-install-location*`/conf/httpd.conf`.

Note – Ensure that your configuration of users and groups meets the security requirements for this directory. For example, to restrict access to this directory, add the Apache user to the same user group as the owner of the directory.

---

a. **To ensure that the Auto Apply feature operates correctly, grant the Apache user read access, write access, and execute access to the** *apache-install-location*`/conf/` **directory.**

   - **If the Apache user is in the same group as the owner of this directory, change the mode to 775.**
   - **If the Apache user is in a different group than the owner of this directory, change the mode to 777.**

b. **To ensure that the load balancer plug-in is initialized when Apache is started, grant the Apache user read access and write access to the following files:**

   - *apache-install-location*`/conf/loadbalancer.xml`
   - *apache-install-location*`/conf/sun-loadbalancer_1_2.dtd`

Exporting and Importing the DAS Certificate

You must manually export the DAS certificate using the following command:

appserver-install-dir`/lib/upgrade/certutil -L -d` *appserver-instance-dir*`/config -n s1as -a -o sjsas.crt`

This certificate will be required at the time of installing the load balancer plug-in.

The Application Server 9.1 installation program performs the following tasks for you.

- Imports the DAS certificate by copying `sjsas.crt` to the *apache-install-dir*/conf/ssl.crt directory.
- Appends the following lines to `httpd.conf`.

```
<Location /lbconfigupdate>
SSLVerifyClient require
SSLVerifyDepth 1
SSLRequireSSL
SSLCACertificateFile apache-install-dir//conf/ssl.crt/sjsas.crt
SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
and %{SSL_CLIENT_S_DN_O} eq "Sun Microsystems" \
and %{SSL_CLIENT_S_DN_OU} eq "Sun Java System Application Server" \
and %{SSL_CLIENT_M_SERIAL} eq "<*serial number*>" )
</Location>
<Location /getmonitordata>
SSLVerifyClient require
SSLVerifyDepth 1
SSLRequireSSL
SSLCACertificateFile apache-install-dir/conf/ssl.crt/sjsas.crt
SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
and %{SSL_CLIENT_S_DN_O} eq "Sun Microsystems" \
   and %{SSL_CLIENT_S_DN_OU} eq "Sun Java System Application Server" \
   and %{SSL_CLIENT_M_SERIAL} eq "<*serial number*>" )
   </Location>
```

Modifications Made by the HTTP Load Balancer Plug-in Installer

The load balancer plug-in installation program extracts the necessary files to the `modules` directory in the web server's root directory:

It adds the following entries to the web server instance's `httpd.conf` file:

```
##BEGIN EE LB Plugin Parameters
LoadModule apachelbplugin_module modules/mod_loadbalancer.so
#AddModule mod_apachelbplugin.cpp
<IfModule mod_apachelbplugin.cpp>
  config-file webserver-instance/httpd/conf/loadbalancer.xml
  locale en
</IfModule>
<VirtualHost machine-ip-address>
  DocumentRoot "webserver-instance/httpd/htdocs"
  ServerName server-name
</VirtualHost>
##END EE LB Plugin Parameters
```

Configuring Apache After Installing the HTTP Load Balancer Plug-In

Apache Web Server must have the correct security files to work with the load balancer plug-in. The load balancer depends on the NSS (Network Security Service) library, which requires these security database files. You need to get these security database files from Communications Application Server, so an installation of Communications Application Server must be available in a location accessible by the Web Server.

To configure Apache security files to work with the load balancer, do the following:

Append `/usr/lib/mps` to `LD_LIBRARY_PATH` in the *Apache-install-dir*`/bin/apachectl` script.

▼ To Create a Security Certificate for Apache

These steps are required to support HTTPS requests on Apache.

For detailed information on setting up a security certificate on Apache, see the instructions on `http://httpd.apache.org/docs/2.2/ssl/ssl_faq.html` and `http://www.modssl.org/docs/2.8/ssl_faq.html`. The following procedure is adapted from those web sites.

1. **Set up the following environment variable:**

   `OPENSSL_CONF=`*OpenSSL-installation-directory*`/apps/openssl.cnf`.

2. **Create the server certificate and key by executing the following command:**

   ```
   openssl req -new -x509 -keyout newreq.pem -out newreq.pem -days 365
   ```

   When asked for a common name, give the host name on which you plan to run Apache. For all other prompts, enter values that meet any specific requirements you have.

   This command creates `newreq.pem`.

3. **Open the newly-created `newreq.pem` from the location where the `openssl` command was run.**

4. **Copy the lines beginning with BEGIN CERTIFICATE and ending with END CERTIFICATE and paste them in** *Apache-install-dir*`/conf/ssl.crt/server.crt`**. For example:**

   ```
   -----BEGIN CERTIFICATE-----
   ....
   ...
   -----END CERTIFICATE-----
   ```

5. **Copy the lines beginning with BEGIN RSA PRIVATE KEY and END RSA PRIVATE KEY and paste them in** *Apache-install-dir*`/conf/ssl.key/server.key`**. For example:**

   ```
   -----BEGIN RSA PRIVATE KEY-----
   ...
   ...
   ...
   -----END RSA PRIVATE KEY-----
   ```

6. **Make sure that the variables `SSLCertificateKeyFile`and `SSLCertificateFile` in** *Apache-install-dir*`/conf/ssl.conf` **have the correct values.**

7. **Ensure that the ServerName is not www.example.com. The ServerName should be the actual host name where Apache will run, matching the Common Name you entered when creating the server certificate and key.**

Modifying httpd.conf parameters to enable sticky round robin

For the sticky round robin feature to work, in the `httpd.conf` file, under the section `prefork MPM`, ensure that the values of the parameters `StartServers` and `maxclients` are set to 1.

Otherwise, every new session request will spawn a new Apache process and the load balancer plug-in will be initialized resulting in requests landing in the same instance.

Starting Apache on Solaris and Linux

In general, you should start Apache with the same user that installed the Communications Application Server. You must start Apache as root under the following circumstances:

- If you are a Java Enterprise System user.
- If you've used port numbers which are less than 1024.
- If Apache runs as a different user from the user that starts it.

To start Apache in SSL mode, use one of the following commands:

`apachetl startssl` or **`apachetl -k start -DSSL`**

If needed, check the Apache web site for the latest information on starting the Apache server.

Verifying the Setup

1. Install the load balancer plug-in. For detailed steps to install the plug-in, see *Sun Java System Application Server 9.1 Installation Guide*. During the installation, provide the path to the DAS certificate.
2. Log in to the Application Server Admin Console and create a new cluster. For steps to create a new cluster, refer to the Admin Console Online Help.
3. Create a new HTTP Load Balancer. While creating the load balancer, specify the web server host as the device host, web server SSL Port as the device port and select the cluster you created in the previous step as the target. For detailed steps to create a new HTTP Load Balancer, refer to the Admin Console Online Help.
4. To verify that the communication between the DAS and the web server is working properly, in the Admin Console, navigate to the HTTP Load Balancers node and click the HTTP Load Balancer. In the Load Balancer Device Settings page that appears, press the Test Connection button.

   If you have not enabled the Automatically Apply Changes option while creating a load balancer, then you must manually export the load balancer configuration by going to the Export tab and clicking Apply Changes now.
5. If the test connection fails, be sure to check the Application Server domain logs and the web server logs to troubleshoot the problem. Also check if all the configuration steps have been performed correctly.

Using Microsoft IIS

To use Microsoft Internet Information Services (IIS) with the load balancer plug-in, follow the steps provided in these sections.

▼ To Configure Microsoft IIS to use the HTTP Load Balancer Plug-in

1. **Open the Internet Services Manager.**
2. **Select the web site for which you want to enable the plug-in.**

   This web site is typically named the Default Web Site.
3. **Right click on the web site and select Properties to open the Properties notebook.**
4. **Add a new ISAPI filter, following these steps:**
   a. **Open the ISAPI Filters tab.**

   b. **Click Add.**

   c. **In the Filter Name field, enter `Communications Application Server`**

   d. **In the Executable field, type `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll`.**

   e. **Click OK, and close the Properties notebook.**
5. **Create and configure a new virtual directory:**
   a. **Right click on the default web site, select New, and then Virtual Directory.**

      The Virtual Directory Creation Wizard opens.

   b. **In the Alias field, type `sun-passthrough`.**

   c. **In the Directory field, type `C:\Inetpub\wwwroot\sun-passthrough`.**

   d. **Check the Execute Permission checkbox.**

      Leave all other permission-related check boxes are left unchecked.

   e. **Click Finish.**

6 **Add the path of the `sun-passthrough.dll` file, the Communications Application Server *as-install*`/bin` and the Communications Application Server *as-install*`/lib` to the system's `PATH` environment variable.**

7 **For IIS 6.0 users, configure the Load Balancer Web Service Extension to run in IIS 6 using the following steps:**

a. **In the IIS manager, expand the local computer, and click Web Service Extensions.**

b. **In the Tasks pane, select Add a new Web Service Extension.**

c. **Enter the name of the Extension as `Sun-Passthrough` and click Add.**

d. **Type the path to `sun-passthrough.dll`, `C:\Inetpub\wwwroot\sun-passthrough`.**

e. **Click OK.**

f. **Select Set extension status to Allowed.**

8 **For IIS 6.0 users, create the file `C:\inetput\wwwroot\sun-passthrough\lb.log` and give NTFS write and modify permissions to the group `IIS_WPG` on the file.**

Because IIS 6.0 runs in Worker Process Isolation Mode, it runs the IIS server with the security privileges of the group `IIS_WPG`.

9 **For all IIS users, restart the machine.**

10 **Verify that the web server, load balancer plug-in, and Communications Application Server are operating correctly.**

Type the following in a web browser to access the web application context root: **`http://`***web-server-name***`/`***web-application*, where *web-server-name* is the host name or IP address of the web server and *web-application* is the context root that you listed in the `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` file.

Tip – The ISAPI filter status should be green. To check the filter status, access the web site's Properties notebook and click the ISAPI Filters tab. If the status is not green, try sending any HTTP request to the IIS HTTP port. It is OK if the request fails. Recheck the ISAPI filter status.

Automatically configured sun-passthrough properties

The installer automatically configures the following properties in `sun-passthrough.properties`. You can change the default values.

Property	Definition	Default Value
`lb-config-file`	Path to the load balancer configuration file	*IIS-www-root*`\sun-passthrough\loadbalancer.xml`
`log-file`	Path to the load balancer log file	*IIS-www-root*`\sun-passthrough\lb.log`
`log-level`	Log level for the web server	`INFO`

Note – The Auto Apply feature of Application Server 9.1 is not currently supported with IIS.

CHAPTER 4

Configuring HTTP Load Balancing

This chapter describes the HTTP load balancer plug-in. It includes the following topics:

For information on other types of load balancing, see Chapter 2, "Configuring Converged Load Balancing," Chapter 9, "Java Message Service Load Balancing and Failover"and Chapter 10, "RMI-IIOP Load Balancing and Failover."

This section discusses using the HTTP load balancing plug-in included with the Communications Application Server. Another HTTP load balancing option is to use the Sun Secure Application Switch with the Communications Application Server for a hardware-based load balancing solution. For a tutorial on configuring this solution, see the article Clustering and Securing Web Applications: A Tutorial (`http://developers.sun.com/prodtech/appserver/reference/techart/load-balancing.html`).

How the HTTP Load Balancer Works

The load balancer attempts to evenly distribute the workload among multiple Application Server instances (either stand-alone or clustered), thereby increasing the overall throughput of the system.

The HTTP Load Balancer enables high availability of services deployed on Java EE Application Servers. While doing so, it fails over a session request to another server instance if the original servicing instance is detected to be unavailable or unhealthy to service a request. For HTTP session information to persist, you must be using the Cluster profile and have configured HTTP session persistence using in-memory replication. For more information, see Chapter 8, “Configuring High Availability Session Persistence and Failover.”

Note – The load balancer does not handle URI/URLs that are greater than 8k.

HTTP Load Balancing Algorithm

The Sun Java System Application Server load balancer, by default, uses a *sticky round robin algorithm* to load balance incoming HTTP and HTTPS requests.

When a new HTTP request is sent to the load balancer plug-in, it is forwarded to an application server instance based on a simple round robin scheme. If the request is for a session-based application, then this also includes a request for a new session. Subsequent requests from the same client for the same session-based application are considered assigned or sticky requests and are routed by the load balancer to the same instance. Hence, the name sticky round robin. Requests to a non session-based application and the first request for a session-based application are called unassigned requests. Stickiness is achieved by using cookies, or explicit URL rewriting. The load balancer determines the method of stickiness automatically.

The load balancer plug-in uses the following methods to determine session stickiness:

- **Cookie Method**: the load balancer plug-in uses a separate cookie to record the route information. The HTTP client (typically, the web browser) must support cookies to use the cookie based method. If the HTTP client is unable to accept cookies, the plug-in uses the following method.
- **Explicit URL Rewriting**: the sticky information is appended to the URL. This method works even if the HTTP client does not support cookies.

From the sticky information, the load balancer plug-in first determines the instance to which the request was previously forwarded. If that instance is found to be healthy, the load balancer plug-in forwards the request to that specific application server instance. Therefore, all requests for a given session are sent to the same application server instance.

What's New in the HTTP Load Balancer Plug-in

In Sun Java System Application Server 9.1, the functionality of the load balancer is enhanced to provide increased flexibility and ease-of-use through the following features.

Auto Apply

Application Server allows changes to the load balancer configuration made from the Admin Console to be automatically sent over the wire to the Web Server configuration directory. With previous versions of Application Server, the load balancer configuration had to be exported and then copied over to the web server configuration directory.

Weighted Round Robin

The load balancer enables improved distribution of HTTP requests. The administrator can use an attribute called 'weight' to specify how requests will be proportionately routed to an instance. For example, suppose a cluster has two instances, and the administrator has assigned a weight of 100 to instance x and a weight of 400 to instance y. Now, for every 100 requests, 20 will go to instance x and 80 will go to instance y.

User–defined Load Balancing

Application Server enables the administrator to define a custom policy for distributing HTTP requests. A custom policy defines the load balancing algorithm that the load balancer plug-in must use. In other words, the Administrator can define which Application Server instance will handle an HTTP request. To use this feature, the administrator needs to develop a shared library, which must implement an interface called `loadbalancer.h`. The shared library can, for example, be used to evaluate the headers of incoming requests provided to it and in accordance to some criteria, select the instance that can serve the request. This shared library would be loaded by the load balancer.

The interface `loadbalancer.h` is available under `webserver_install_dir/plugins/lib/install`. Application Server also bundles a shared library called `roundrobin.c`, that the administrator can use as a template to build the shared library.

Setting Up HTTP Load Balancing

This section describes how to set up the load balancer plug-in and includes the following sections:

- "Prerequisites for Setting Up HTTP Load Balancing" on page 73
- "Procedure to Set Up HTTP Load Balancing" on page 74
- "HTTP Load Balancer Deployments" on page 77

Prerequisites for Setting Up HTTP Load Balancing

Before configuring your load balancer, you must:

- Install a supported web server and configure it. For more information on configuring a supported web server, see Chapter 3, "Configuring Web Servers for HTTP Load Balancing."
- Install the load balancer plug-in.

  For information on the installation procedure, see *Sun Java System Application Server 9.1 Installation Guide*.
- Create Communications Application Server clusters or server instances to participate in load balancing.

- Deploy applications to these clusters or instances.

Note – If you have a deployment scenario where the Application Server instances and the load balancer are installed on different network domains, then you must create the node agent by specifying the fully qualified domain name using the option, `--agentproperties`. For example, `asadmin create-node-agent --agentproperties remoteclientaddress=myhost.country.example.com test-na`.

Procedure to Set Up HTTP Load Balancing

Use the Admin Console GUI or the `asadmin` tool to configure load balancing in your environment. The following sections provide you more information.

▼ To Set Up Load Balancing Using the Admin Console

1 **Create a load balancer configuration.**

In the Admin Console, on the left frame, click HTTP Load Balancers and then click New. In the New HTTP Load Balancer page, provide the device details and also select the target cluster or instances.

2 **Add a reference to a cluster or stand-alone server instance for the load balancer to manage.**

To do this using the Admin Console, on the left frame, click the HTTP Load Balancers node and then click the desired load balancer listed under the node. Open the Targets tab, click Manage Targets and in the Manage Targets page, select the required target.

If you created the load balancer configuration with a target, and that target is the only cluster or stand-alone server instance the load balancer references, skip this step.

3 **Enable the cluster or stand-alone server instance referenced by the load balancer.**

To enable a stand-alone server instance using the Admin Console, on the left frame, click the HTTP Load Balancers node and then click the desired load balancer listed under the node. Open the Targets tab and in the Targets table, click the check box next to the instance you want to enable and click Enable.

To enable a server instance in a cluster, select the load balancer as explained above and in the Targets tab, click the desired cluster. Now open the Instances tab, select the desired instance, and from the Load Balancer Actions drop down list, select Enable Load Balancing.

The equivalent command to enable a cluster or a stand-alone instance is `asadmin enable-http-lb-server`.

4 **Enable applications for load balancing.**

To do this using the Admin Console, open the Targets tab as mentioned above and click the required cluster. Now, open the Applications tab, select the required application and from the More Actions drop-down list, and select Load Balancer Enable.

These applications must already be deployed and enabled for use on the clusters or stand-alone instances that the load balancer references. Enabling applications for load balancing is a separate step from enabling them for use.

5 **Create a health checker.**

To do this using the Admin Console, open the Targets tab for the load balancer as mentioned in the previous step and in the Targets table, click Edit Health Checker.

The health checker monitors unhealthy server instances so that when they become healthy again, the load balancer can send new requests to them.

Note – If you are using Sun Java System Web Server (6.1 or 7.0), instead of performing steps 6 and 7, you can generate the load balancer configuration file and send the data over the wire to the Web Server in a single step.

To do this using the Admin Console, click the desired load balancer and then open the Export tab. In this tab, click Apply Changes Now. This sends the data to the Web Server configuration directory.

---

6 **Generate the load balancer configuration file.**

To do this using the Admin Console, click the load balancer and then open the Export tab. In this tab, click Export Now.

This command generates a configuration file to use with the load balancer plug-in shipped with the Sun Java System Communications Application Server.

7 **Copy the load balancer configuration file to your web server `config` directory where the load balancer plug-in configuration files are stored.**

Note – To generate the load balancer configuration file automatically and send the data over the wire to the Web Server in a single step, you need to configure Web server for SSL setup and import the DAS certificate. For information on configuring Sun Java System Web Server, see "Configuring Sun Java System Web Server" on page 52.

▼ To Set Up Load Balancing Using the `asadmin` Tool

1 **Create a load balancer configuration.**

To do this, use the command, `asadmin create-http-lb-config`.

Note – You can perform all the following steps (Step 2 through Step 7) using a single `asadmin` command, `create-http-lb` and its options. For more information about this command, see `create-http-lb`(1).

---

2 **Add a reference to a cluster or stand-alone server instance for the load balancer to manage.**

To do this, use the command, `asadmin create-http-lb-ref`. For more information about this command, see `create-http-lb-ref`(1).

If you created the load balancer configuration with a target, and that target is the only cluster or stand-alone server instance the load balancer references, skip this step.

3 **Enable the cluster or stand-alone server instance referenced by the load balancer.**

To do this, use the command `asadmin enable-http-lb-server`. For more information about this command, see `enable-http-lb-server`(1).

4 **Enable applications for load balancing.**

To do this, use the command `asadmin enable-http-lb-application`. For more information about this command, see `enable-http-lb-application`(1).

These applications must already be deployed and enabled for use on the clusters or stand-alone instances that the load balancer references. Enabling applications for load balancing is a separate step from enabling them for use.

5 **Create a health checker.**

To do this, use the command, `asadmin create-http-health-checker`. For more information about this command, see `create-http-health-checker`(1).

The health checker monitors unhealthy server instances so that when they become healthy again, the load balancer can send new requests to them.

Note – If you are using Sun Java System Web Server (6.1 or 7.0), instead of performing steps 6 and 7, you can generate the load balancer configuration file and send the data over the wire to the Web Server in a single step.

To do this using the `asadmin` tool, set the `--autoapplyenabled` option of the `create-http-lb` command to `true`. For more information about this command, see `create-http-lb`(1).

---

6 **Generate the load balancer configuration file.**

To do this, use the command `asadmin export-http-lb-config`.

This command generates a configuration file to use with the load balancer plug-in shipped with the Sun Java System Communications Application Server.

7 **Copy the load balancer configuration file to your web server `config` directory where the load balancer plug-in configuration files are stored.**

Note – To generate the load balancer configuration file automatically and send the data over the wire to the Web Server in a single step, you need to configure Web server for SSL setup and import the DAS certificate. For information on configuring Sun Java System Web Server, see "Configuring Sun Java System Web Server" on page 52.

HTTP Load Balancer Deployments

You can configure your load balancer in different ways, depending on your goals and environment, as described in the following sections:

- "Using Clustered Server Instances" on page 77
- "Using Multiple Stand-Alone Instances" on page 77

Using Clustered Server Instances

The most common way to deploy the load balancer is with a cluster or clusters of server instances. By default all the instances in a cluster have the same configuration and the same applications deployed to them. The load balancer distributes the workload between the server instances and requests fail over from an unhealthy instance to a healthy one. If you've configured HTTP session persistence, session information persists when the request is failed over.

If you have multiple clusters, requests can be load balanced across clusters but are only failed over between the instances in a single cluster. Use multiple clusters in a load balancer to easily enable rolling upgrades of applications. For more information, see "Upgrading Applications Without Loss of Availability" on page 91.

Note – Requests cannot be load balanced across clusters and stand-alone instances.

Using Multiple Stand-Alone Instances

It is also possible to configure your load balancer to use multiple stand-alone instances, and load balance and failover requests between them. However, in this configuration, you must manually ensure that the stand-alone instances have homogenous environments and the same applications deployed to them. Because clusters automatically maintain a homogenous environment, for most situations it is better and easier to use clusters.

Configuring the HTTP Load Balancer

Load balancer configuration is maintained in the `domain.xml` file. Configuring a load balancer is extremely flexible:

- A load balancer services only one domain, though a domain can have multiple load balancers associated with it.
- Each load balancer configuration can have multiple load balancers associated with it, though each load balancer has only one load balancer configuration.

These sections describe, in more detail, how to create, modify, and use a load balancer configuration:

Configuring an HTTP Load Balancer on the DAS

In Application Server 9.1, you can create a load balancer configuration on the DAS using the Admin Console or the `asadmin` command `create-http-lb`. The following steps explain how you to do that. If you want more information about the asadmin commands, see the man pages or the Reference Manual for `create-http-lb`, `delete-http-lb`, and `list-http-lbs`.

In the Admin Console, scroll down the left frame, click the HTTP Load Balancers node and then in the HTTP Load balancers page on the right, click New. In the New HTTP Load Balancer page, provide the following details of the machine hosting the load balancer.

Field	Description
Name	A name for the load balancer configuration.

Enabled	Click the Enabled check box to automatically push the load balancer configuration changes to the physical load balancer residing in the web server configuration directory.
Host	The server on which the web server instance is installed.
Admin Port	The secure HTTP listener port.
Proxy Host	The server on which the proxy server instance is installed.
Proxy Port	The port number used by the proxy server.

You can also create a load balancer configuration using the `asadmin` command `create-http-lb-config`. Table 4–1 describes the parameters. For more information see the documentation for `create-http-lb-config`, `delete-http-lb-config`, and `list-http-lb-configs`.

TABLE 4–1 Load Balancer Configuration Parameters

Parameter	**Description**
response timeout	Time in seconds within which a server instance must return a response. If no response is received within the time period, the server is considered unhealthy. The default is 60.
HTTPS routing	Whether HTTPS requests to the load balancer result in HTTPS or HTTP requests to the server instance. For more information, see "Configuring HTTPS Routing" on page 86.
reload interval	Interval between checks for changes to the load balancer configuration file `loadbalancer.xml`. When the check detects changes, the configuration file is reloaded. A value of 0 disables reloading. For more information, see "Enabling Dynamic Reconfiguration" on page 84.
monitor	Whether monitoring is enabled for the load balancer.
routecookie	Name of the cookie the load balancer plug-in uses to record the route information. The HTTP client must support cookies. If your browser is set to ask before storing a cookie, the name of the cookie is JROUTE.
target	Target for the load balancer configuration. If you specify a target, it is the same as adding a reference to it. Targets can be clusters or stand-alone instances.

Creating an HTTP Load Balancer Reference

When you create a reference in the load balancer to a stand-alone server or cluster, the server or cluster is added to the list of target servers and clusters the load balancer controls. The referenced server or cluster still needs to be enabled before requests to it are load balanced. If you created the load balancer configuration with a target, that target is already added as a reference.

To create a reference using the Admin Console, on the left frame, click the HTTP Load Balancers node and then click the desired load balancer listed under the node. Open the Targets tab, click Manage Targets and in the Manage Targets page, select the required target. Alternatively, you can create a reference using `create-http-lb-ref`. You must supply the load balancer configuration name and the target server instance or cluster.

To delete a reference, use `delete-http-lb-ref`. Before you can delete a reference, the referenced server or cluster must be disabled using `disable-http-lb-server`.

For more information, see the documentation for `create-http-lb-ref` and `delete-http-lb-ref`.

Enabling Server Instances for HTTP Load Balancing

After creating a reference to the server instance or cluster, enable the server instance or cluster using `enable-http-lb-server`. If you used a server instance or cluster as the target when you created the load balancer configuration, you must enable it. To do this using the Admin Console, on the left frame, click the HTTP Load Balancers node and then click the desired load balancer listed under the node. Now, open the Targets tab and in the Targets table, click the check box next to the instance you want to enable and click Enable.

For more information, see the documentation for `enable-http-lb-server`.

Enabling Applications for HTTP Load Balancing

All servers managed by a load balancer must have homogenous configurations, including the same set of applications deployed to them. Once an application is deployed and enabled for access (this happens during or after the deployment step) you must enable it for load balancing. If an application is not enabled for load balancing, requests to it are not load balanced and failed over, even if requests to the servers the application is deployed to are load balanced and failed over.

When enabling the application, specify the application name and target. If the load balancer manages multiple targets (for example, two clusters), enable the application on all targets.

To enable an application using the Admin Console, on the left frame, click the HTTP Load Balancers node and then click the desired load balancer listed under the node. Open the Targets tab as mentioned above and click the required cluster. Now, open the Applications tab, select the required application and from the More Actions drop-down list, and select Load Balancer Enable. If you want to do this from the command line, you can use the command `asadmin enable-http-lb-application`. For more information about the command, see the man pages.

If you deploy a new application, you must also enable it for load balancing and export the load balancer configuration again.

Creating the HTTP Health Checker

The load balancer's health checker periodically checks all the configured Communications Application Server instances that are marked as unhealthy. A health checker is not required, but if no health checker exists, or if the health checker is disabled, the periodic health check of unhealthy instances is not performed. The load balancer will not be able to determine when an unhealthy instance becomes healthy.

The load balancer's health check mechanism communicates with the application server instance using HTTP. The health checker sends an HTTP request to the URL specified and waits for a response. A status code in the HTTP response header between 100 and 500 means the instance is healthy.

Note – If you have a deployment scenario where the load balancer is the front–end for a cluster that has instances using a secured port with client certificate authentication enabled, the health checker will not be able to perform a health check of the instances. Hence, those instances will always be marked unhealthy and no requests will be sent to them.

Creating a Health Checker

To specify the health checker properties, you can use the Admin Console or the `asadmin create-http-health-checker` command. To do this in the Admin Console, navigate to the HTTP Load Balancers node, expand it and select the load balancer. Then, open the Target tab, and in the Targets table, click the Edit Health Checker link for the desired target. Specify the following parameters.

TABLE 4–2 Health Checker Parameters

Parameter	Description	Default
Load Balancer	Click the Enabled check box to make the selected server available for load balancing.	False/Disabled
Disable Timeout	The amount of time, in minutes, this server takes to reach a quiescent state after having been disabled .	30 minutes
url	Specifies the listener's URL that the load balancer checks to determine its state of health.	"/"
interval	Specifies the interval in seconds at which health checks of instances occur. Specifying 0 disables the health checker.	30 seconds
timeout	Specifies the timeout interval in seconds within which a response must be obtained for a listener to be considered healthy.	10 seconds

If an application server instance is marked as unhealthy, the health checker polls the unhealthy instances to determine if the instance has become healthy. The health checker uses the specified URL to check all unhealthy application server instances to determine if they have returned to the healthy state.

If the health checker finds that an unhealthy instance has become healthy, that instance is added to the list of healthy instances.

For more information see the documentation for `create-http-health-checker` and `delete-http-health-checker`.

Additional Health Check Properties for Healthy Instances

The health checker created by `create-http-health-checker` only checks unhealthy instances. To periodically check healthy instances, set some additional properties in your exported `loadbalancer.xml` file.

Note – These properties can only be set by manually editing `loadbalancer.xml` *after* you've exported it. There is no equivalent `asadmin` command to use.

To check healthy instances, set the following properties.

TABLE 4–3 Health-checker Manual Properties

Property	Definition
active-healthcheck-enabled	True/false flag indicating whether to ping healthy server instances to determine whether they are healthy. To ping server instances, set the flag to true.
number-healthcheck-retries	Specifies how many times the load balancer's health checker pings an unresponsive server instance before marking it unhealthy. Valid range is between 1 and 1000. A default value to set is 3.

Set the properties by editing the `loadbalancer.xml` file. For example:

```
<property name="active-healthcheck-enabled" value="true"/>
<property name="number-healthcheck-retries" value="3"/>
```

If you add these properties, then subsequently edit and export the `loadbalancer.xml` file again, the newly exported configuration won't contain these properties. You must add these properties again to the newly exported configuration.

Exporting the HTTP Load Balancer Configuration File

The load balancer plug-in available with Sun Java System Communications Application Server uses a configuration file called `loadbalancer.xml`. After configuring the load balancer, you can export the configuration details from `domain.xml` to the `loadbalancer.xml` file. You can do this using the Admin Console or the asadmin utility.

▼ To Export the Load Balancer Configuration Using the Admin Console

1. **Navigate to the HTTP Load Balancers node and expand it.**

2. **Click the desired load balancer.**

   All the load balancer configuration details are displayed in the General, Settings and Target tabs.

3. **Open the Export tab and click Export now.**

4. **Copy the exported load balancer configuration file to the web server's configuration directory.**

▼ To Export the Load Balancer Configuration Using the asadmin tool

1. **Export a `loadbalancer.xml` file using the `asadmin` command `export-http-lb-config`.**

   Export the `loadbalancer.xml` file for a particular load balancer configuration. You can specify a path and a different file name. If you don't specify a file name, the file is named `loadbalancer.xml.` *load-balancer-config-name*. If you don't specify a path, the file is created in the *domain-dir*/`generated` directory.

   To specify a path on Windows, use quotes around the path. For example, `"C:\Sun\AppServer\loadbalancer.xml"`.

2. **Copy the exported load balancer configuration file to the web server's configuration directory.**

   For example, for the Sun Java System Web Server, that location usually is *web-server-root*/`config`.

   The load balancer configuration file in the web server's configuration directory must be named `loadbalancer.xml`. If your file has a different name, such as `loadbalancer.xml.` *load-balancer-config-name*, you must rename it.

Changing the HTTP Load Balancer Configuration

If you change a load balancer configuration by creating or deleting references to servers, deploying new applications, enabling or disabling servers or applications, and so on, export the load balancer configuration file again and copy it to the web server's `config` directory. For more information, see "Exporting the HTTP Load Balancer Configuration File" on page 83

The load balancer plug-in checks for an updated configuration periodically based on the reload interval specified in the load balancer configuration. After the specified amount of time, if the load balancer discovers a new configuration file, it starts using that configuration.

Enabling Dynamic Reconfiguration

With dynamic reconfiguration, the load balancer plug-in periodically checks for an updated configuration.

To enable dynamic reconfiguration:

- When creating a load balancer configuration, use the `--reloadinterval` option with `asadmin create-http-lb` .

  This option sets the amount of time between checks for changes to the load balancer configuration file `loadbalancer.xml`. A value of 0 disables dynamic reconfiguration. By default, dynamic reconfiguration is enabled, with a reload interval of 60 seconds.
- If you have previously disabled it, or to change the reload interval, use the `asadmin set` command.

  After changing the reload interval, export the load balancer configuration file again and copy it to the web server's `config` directory, then restart the web server.

Note – If the load balancer encounters a hard disk read error while attempting reconfiguration, it uses the configuration that is currently in memory. The load balancer also ensures that the modified configuration data is compliant with the DTD before over writing the existing configuration.

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

The error log for Sun Java System Web Server' is at:
web-server-install-dir`/`*web-server-instance*`/logs/`.

Disabling (Quiescing) a Server Instance or Cluster

Before stopping an application server for any reason, the instance should complete serving requests. The process of gracefully disabling a server instance or cluster is called quiescing.

The load balancer uses the following policy for quiescing application server instances:

- If an instance (either stand-alone or part of a cluster) is disabled, and the timeout has not expired, sticky requests continue to be delivered to that instance. New requests, however, are not sent to the disabled instance.
- When the timeout expires, the instance is disabled. All open connections from the load balancer to the instance are closed. The load balancer does not send any requests to this instance, even if all sessions sticking to this instance were not invalidated. Instead, the load balancer fails over sticky requests to another healthy instance.

▼ To disable a server instance or cluster

1. **Run** `asadmin disable-http-lb-server`**, setting the timeout (in minutes).**
2. **Export the load balancer configuration file using** `asadmin export-http-lb-config`**.**
3. **Copy the exported configuration to the web server** `config` **directory.**
4. **Stop the server instance or instances.**

Disabling (Quiescing) an Application

Before you undeploy a web application, the application should complete serving requests. The process of gracefully disabling an application is called quiescing. When you quiesce an application, you specify a timeout period. Based on the timeout period, the load balancer uses the following policy for quiescing applications:

- If the timeout has not expired, the load balancer does not forward new requests to the application, but returns them to the web server. However, the load balancer continues to forward sticky requests until the timeout expires.
- When the timeout expires, the load balancer does not accept any requests for the application, including sticky requests.

  When you disable an application from every server instance or cluster the load balancer references, the users of the disabled application experience loss of service until the application is enabled again. If you disable the application from one server instance or cluster while keeping it enabled in another server instance or cluster, users can still access the application. For more information, see "Upgrading Applications Without Loss of Availability" on page 91.

▼ To disable an application

1. **Use** `asadmin disable-http-lb-application`**, specifying the following:**
   - Timeout (in minutes).
   - Name of the application to disable.
   - Target cluster or instance on which to disable it.
2. **Export the load balancer configuration file using** `asadmin export-http-lb-config`**.**
3. **Copy the exported configuration to the web server** `config` **directory.**

Configuring HTTP and HTTPS Failover

The load balancer plug-in fails over HTTP/HTTPS sessions to another application server instance if the original application server instance to which the session was connected becomes unavailable. This section describes how to configure the load balancer plug-in to enable HTTP/HTTPS routing and session failover.

HTTPS Routing

The load balancer plug-in routes all incoming HTTP or HTTPS requests to application server instances. However, if HTTPS routing is enabled, an HTTPS request will be forwarded by the load balancer plug-in to an application server using an HTTPS port only. HTTPS routing is performed on both new and sticky requests.

If an HTTPS request is received and no session is in progress, then the load balancer plug-in selects an available application server instance with a configured HTTPS port, and forwards the request to that instance.

In an ongoing HTTP session, if a new HTTPS request for the same session is received, then the session and sticky information saved during the HTTP session is used to route the HTTPS request. The new HTTPS request is routed to the same server where the last HTTP request was served, but on the HTTPS port.

Configuring HTTPS Routing

The `httpsrouting` option of the `create-http-lb-config` command controls whether HTTPS routing is turned on or off for all the application servers that are participating in load balancing. If this option is set to false, all HTTP and HTTPS requests are forwarded as HTTP. If set to true, HTTPS are forwarded as HTTPS requests. Set HTTPS routing when creating a new load balancer configuration, or change it later using the `asadmin set` command.

Note –

- For HTTPS routing to work, one or more HTTPS listeners must be configured.
- If `https-routing` is set to `true`, and a new or a sticky request comes in where there are no healthy HTTPS listeners in the cluster, then that request generates an error.

Known Issues

The Load Balancer has the following limitations with HTTP/HTTPS request processing.

- If a session uses a combination of HTTP and HTTPS requests, then the first request must be an HTTP Request. If the first request is an HTTPS request, it cannot be followed by an HTTP request. This is because the cookie associated with the HTTPS session is not returned by the browser. The browser interprets the two different protocols as two different servers, and initiates a new session. This limitation is valid only if `httpsrouting` is set to true.
- If a session has a combination of HTTP and HTTPS requests, then the application server instance must be configured with both HTTP and HTTPS listeners. This limitation is valid only if `httpsrouting` is set to `true`.
- If a session has a combination of HTTP and HTTPS requests, then the application server instance must be configured with HTTP and HTTPS listeners that use standard port numbers, that is, 80 for HTTP, and 443 for HTTPS. This limitation applies regardless of the value set for `httpsrouting`.

Using Redirects with the HTTP Load Balancer

Use redirects to redirect a request from one URL to another URL. For example, use redirects to send users to a different web site (for example, redirecting from an old version of an application to a newer version) or from HTTP to HTTPS or from HTTPS to HTTP. Redirects can be enabled in a number of ways in the application (for example, servlet-based redirects, `web.xml` redirects). However, sending a redirect URL through the load balancer may require some additional configuration of the Communications Application Server or the load balancer. Note that redirects are different from requests that are forwarded using HTTPS Routing. When using redirects, set `httpsrouting` to false. If configuring HTTPS requests to be forwarded to HTTP, use "HTTPS Routing" on page 86.

The following properties affect redirects: the `authPassthroughEnabled` and `proxyHandler` properties of the HTTP service or HTTP listener, and the `rewrite-location` property in the `loadbalancer.xml` file.

The authPassthroughEnabled Property

When the Communications Application Server `authPassthroughEnabled` property is set to true, information about the original client request (such as client IP address, SSL keysize, and authenticated client certificate chain) is sent to the HTTP listeners using custom request headers. The `authPassThroughEnabled` property allows you to take advantage of a hardware accelerator for faster SSL authentication if you have one installed. It is easier to configure a hardware accelerator on the load balancer than on each clustered Communications Application Server instance.

Caution – Set `authPassthroughEnabled` to true only if the Communications Application Server is behind a firewall.

Use the `asadmin set` command to set the `authPassthroughEnabled` property on the HTTP service or the individual HTTP listener. The setting for the individual HTTP listener takes precedence over the setting for the HTTP service.

To set the `authPassthroughEnabled` property on all HTTP/HTTPS listeners, use the following command:

`asadmin set`
*cluster-name***`-config.http-service.property.authPassthroughEnabled=true`**

To set it on an individual listener, use the following command:

`asadmin set` *cluster-name***`-config.http-service.http-listener.`***listener-name***`.property.`**
`authPassthroughEnabled=true`

The proxyHandler Property

The proxy handler for the Communications Application Server is responsible for retrieving information about the original client request that was intercepted by a proxy server (in this case, a load balancer) and forwarded to the Application Server, and for making this information available to the web application (deployed on the Application Server) that is the target of the client request. If the intercepting proxy server is SSL-terminating, the proxy handler retrieves and makes available additional information about the original request, such as whether the original request was an HTTPS request, and whether SSL client authentication is enabled. Use the `proxyHandler` property only if `authPassThroughEnabled` is set to true.

The proxy handler inspects incoming requests for the custom request headers through which the proxy server conveys the information about the original client request, and makes this information available to the web application on the Application Server using standard `ServletRequest` APIs.

The proxy handler implementation is configurable, either globally at the HTTP service level or for individual HTTP listeners, with the `proxyHandler` property, whose value specifies the fully-qualified class name of an implementation of the `com.sun.appserv.ProxyHandler` abstract class. Configurable proxy handler implementations allow the Application Server to work with any proxy server, as long as the proxy handler implementation knows about the HTTP request header names, and understands the format of their values, through which the proxy server conveys information about the original client request.

The proxy handler for the Communications Application Server reads and parses the SSL certificate chain from the request header. This allows a back-end application server instance to retrieve information about the original client request that was intercepted by an SSL-terminating proxy server (in this case, a load balancer). You can use the default proxy handler settings, or configure your own using the `proxyHandler` property of the HTTP service or HTTP/HTTPS listener. The `proxyHandler` property specifies the fully-qualified class name of a custom implementation of the `com.sun.appserv.ProxyHandler` abstract class used by the listener or all listeners.

An implementation of this abstract class inspects a given request for the custom request headers through which the proxy server communicates the information about the original client request to the Application Server instance, and returns that information to its caller. The default implementation reads the client IP address from an HTTP request header named `Proxy-ip`, the SSL keysize from an HTTP request header named `Proxy-keysize`, and the SSL client certificate chain from an HTTP request header named `Proxy-auth-cert`. The `Proxy-auth-cert` value must contain the BASE-64 encoded client certificate chain without the BEGIN CERTIFICATE and END CERTIFICATE boundaries and with `\n` replaced with `% d% a`.

You can only use this property if `authPassThroughEnabled` is set to true. If you set the `proxyHandler` property on an individual HTTP or HTTPS listener, it overrides the default setting for all listeners.

Use the `asadmin set` command to set the `proxyHandler` property on the HTTP service or the individual HTTP listener.

To set the `proxyHandler` property on all HTTP/HTTPS listeners, use the following command:

```
asadmin set cluster-name-config.http-service.property.proxyHandler=classname
```

To set it on an individual listener, use the following command:

```
asadmin set cluster-name-config.http-service.http-listener.listener-name.property.proxyHandler=classname
```

The rewrite-location Property

If set to true, the `rewrite-location` property rewrites the original request information and includes the protocol (HTTP or HTTPS), host, and port information By default, the `rewrite-location` property is set to true to maintain backward compatibility with previous Communications Application Server releases.

The `rewrite-location` property is not available through the `asadmin create-http-lb-config` or `asadmin set` commands. To use the property, manually add it to the `loadbalancer.xml` file after exporting your load balancer configuration. For example, add the following to the exported `loadbalancer.xml` file:

```
<property name="rewrite-location" value="false"/>
```

Set the `rewrite-location` property with the following points in mind:

- If `httpsrouting` is false and `authPassthroughEnabled` is not enabled on the Communications Application Server, set the `rewrite-location` property to true. When `authPassthroughEnabled` is not enabled, the Communications Application Server will not be aware of the protocol (HTTP or HTTPS) of the original request. By setting `rewrite-location` to true the load balancer modifies the protocol part of the rewrite location suitably. That is, if the client is sending HTTPS requests, then the load balancer redirects the client to a HTTPS-enabled listener port on the load balancer. The process is the same for HTTP requests.
- If `httpsrouting` is false, and `authPassthroughEnabled` is enabled on the Communications Application Server, then `rewrite-location` can be set to true or false because the Communications Application Server is aware of whether the client request is HTTP or HTTPS. When `authPassthroughEnabled` is enabled, the Communications Application Server modifies the protocol part of rewrite location suitably. If `rewrite-location` is set to false, the load balancer does not rewrite the location of the redirected URL. If set to true, it rewrites the location of the redirected URL. But this rewrite is not needed as the Communications Application Server was aware of HTTPS connections from the client. Also, if the application needs to redirect HTTP to HTTPS or HTTPS to HTTP, you must set the `rewrite-location` parameter to false.

Configuring Idempotent URLs

An *idempotent* request is one that does not cause any change or inconsistency in an application when retried. In HTTP, some methods (such as GET) are idempotent, while other methods (such as POST) are not. Retrying an idempotent URL must not cause values to change on the server or in the database. The only difference is a change in the response received by the user.

Examples of idempotent requests include search engine queries and database queries. The underlying principle is that the retry does not cause an update or modification of data.

To enhance the availability of deployed applications, configure the environment to retry failed idempotent HTTP requests on all the application server instances serviced by a load balancer. This option is used for read-only requests, for example, to retry a search request.

Configure idempotent URLs in the `sun-web.xml` file. When you export the load balancer configuration, idempotent URL information is automatically added to the `loadbalancer.xml` file.

For more information on configuring idempotent URLs, see "Configuring Idempotent URL Requests" in *Sun Java System Application Server 9.1 Developer's Guide*.

Configuring Multiple Web Server Instances

The Sun Java System Application Server installer does not allow the installation of multiple load balancer plug-ins on a single machine. To have multiple web servers with the load balancer plug-in on a single machine, in either a single cluster or multiple clusters, a few manual steps are required to configure the load balancer plug-in.

▼ To Configure Multiple Web Server Instances

1. **Configure the new web server instance to use the load balancer plug-in.**

   For detailed instructions, see "Configuring Sun Java System Web Server" on page 52.

2. **Copy the DTD file `sun-loadbalancer_1_1.dtd` from the existing web server instance's `config` directory to the new instance's `config` directory.**

3. **Set up the load balancer configuration file. Either:**

   - **Copy the existing load balancer configuration.**

     Use an existing load balancer configuration, copy the `loadbalancer.xml` file from the existing web server instance's `config` directory to the new instance's `config` directory.

   - **Create a new load balancer configuration:**

     a. **Use `asadmin create-http-lb-config` to create a new load balancer configuration.**

     b. **Export the new configuration to a `loadbalancer.xml` file using `asadmin export http-lb-config`.**

     c. **Copy that `loadbalancer.xml` file to the new web server's `config` directory.**

        For information on creating a load balancer configuration and exporting it to a `loadbalancer.xml` file, see " Configuring an HTTP Load Balancer on the DAS" on page 78

Upgrading Applications Without Loss of Availability

Upgrading an application to a new version without loss of availability to users is called a *rolling upgrade*. Carefully managing the two versions of the application across the upgrade ensures that current users of the application complete their tasks without interruption, while new users transparently get the new version of the application. With a rolling upgrade, users are unaware that the upgrade occurs.

Application Compatibility

Rolling upgrades pose varying degrees of difficulty depending on the magnitude of changes between the two application versions.

If the changes are superficial, for example, changes to static text and images, the two versions of the application are *compatible* and can both run at once in the same cluster.

Compatible applications must:

- Use the same session information
- Use compatible database schemas
- Have generally compatible application-level business logic
- Use the same physical data source

You can perform a rolling upgrade of a compatible application in either a single cluster or multiple clusters. For more information, see "Upgrading In a Single Cluster" on page 92.

If the two versions of an application do not meet all the above criteria, then the applications are considered *incompatible.* Executing incompatible versions of an application in one cluster can corrupt application data and cause session failover to not function correctly. The problems depend on the type and extent of the incompatibility. It is good practice to upgrade an incompatible application by creating a "shadow cluster" to which to deploy the new version and slowly quiesce the old cluster and application. For more information, see "Upgrading Incompatible Applications" on page 96.

The application developer and administrator are the best people to determine whether application versions are compatible. If in doubt, assume that the versions are incompatible, since this is the safest approach.

Upgrading In a Single Cluster

You can perform a rolling upgrade of an application deployed to a single cluster, providing the cluster's configuration is not shared with any other cluster.

▼ To upgrade an application in a single cluster

1. **Save an old version of the application or back up the domain.**

   To back up the domain use the `asadmin backup-domain` command.

2. **Turn off dynamic reconfiguration (if enabled) for the cluster.**

   To do this with Admin Console:

   a. **Expand the Configurations node.**

b. **Click the name of the cluster's configuration.**

c. **On the Configuration System Properties page, uncheck the Dynamic Reconfiguration Enabled box.**

d. **Click Save**

Alternatively, use this command:

`asadmin set --user` *user* **`--passwordfile`** *password-file*
*cluster-name***`-config.dynamic-reconfiguration-enabled=false`**

3 **Redeploy the upgraded application to the target `domain`.**

If you redeploy using the Admin Console, the domain is automatically the target. If you use `asadmin`, specify the target `domain`. Because dynamic reconfiguration is disabled, the old application continues to run on the cluster.

4 **Enable the redeployed application for the instances using `asadmin enable-http-lb-application`.**

5 **Quiesce one server instance in the cluster from the load balancer.**

Follow these steps:

a. **Disable the server instance using `asadmin disable-http-lb-server`.**

b. **Export the load balancer configuration file using `asadmin export-http-lb-config`.**

c. **Copy the exported configuration file to the web server instance's configuration directory.**

For example, for Sun Java System Web Server, the location is *web-server-install-dir*`/https-`*host-name*`/config/loadbalancer.xml`. To ensure that the load balancer loads the new configuration file, be sure that dynamic reconfiguration is enabled by setting the `reloadinterval` in the load balancer configuration.

d. **Wait until the timeout has expired.**

Monitor the load balancer's log file to make sure the instance is offline. If users see a retry URL, skip the quiescing period and restart the server immediately.

6 **Restart the disabled server instance while the other instances in the cluster are still running.**

Restarting causes the server to synchronize with the domain and update the application.

7 **Test the application on the restarted server to make sure it runs correctly.**

8 **Re-enable the server instance in load balancer.**

Follow these steps:

a. **Enable the server instance using** `asadmin enable-http-lb-server`.

b. **Export the load balancer configuration file using** `asadmin export-http-lb-config`**.**

c. **Copy the configuration file to the web server's configuration directory.**

9 **Repeat steps 5 through 8 for each instance in the cluster.**

10 **When all server instances have the new application and are running, you can enable dynamic reconfiguration for the cluster again.**

Upgrading in Multiple Clusters

▼ To Upgrade a Compatible Application in Two or More Clusters:

1 **Save an old version of the application or back up the domain.**

To back up the domain, use the `asadmin backup-domain` command.

2 **Turn off dynamic reconfiguration (if enabled) for all clusters.**

To do this with Admin Console:

a. **Expand the Configurations node.**

b. **Click the name of one cluster's configuration.**

c. **On the Configuration System Properties page, uncheck the Dynamic Reconfiguration Enabled box.**

d. **Click Save**

e. **Repeat for the other clusters**

Alternatively, use this command:

```
asadmin set --user user --passwordfile password-file
cluster-name-config.dynamic-reconfiguration-enabled=false
```

3 **Redeploy the upgraded application to the target `domain`.**

If you redeploy using the Admin Console, the domain is automatically the target. If you use `asadmin`, specify the target `domain`. Because dynamic reconfiguration is disabled, the old application continues to run on the clusters.

4 **Enable the redeployed application for the clusters using `asadmin enable-http-lb-application`.**

5 **Quiesce one cluster from the load balancer**

a. **Disable the cluster using `asadmin disable-http-lb-server`.**

b. **Export the load balancer configuration file using `asadmin export-http-lb-config`.**

c. **Copy the exported configuration file to the web server instance's configuration directory.**

For example, for Sun Java System Web Server, the location is *web-server-install-dir*`/https-`*host-name*`/config/loadbalancer.xml`. Dynamic reconfiguration must be enabled for the load balancer (by setting the `reloadinterval` in the load balancer configuration), so that the new load balancer configuration file is loaded automatically.

d. **Wait until the timeout has expired.**

Monitor the load balancer's log file to make sure the instance is offline. If users see a retry URL, skip the quiescing period and restart the server immediately.

6 **Restart the disabled cluster while the other clusters are still running.**

Restarting causes the cluster to synchronize with the domain and update the application.

7 **Test the application on the restarted cluster to make sure it runs correctly.**

8 **Enable the cluster in the load balancer:**

a. **Enable the cluster using `asadmin enable-http-lb-server.`**

b. **Export the load balancer configuration file using `asadmin export-http-lb-config`.**

c. **Copy the configuration file to the web server's configuration directory.**

9 **Repeat steps 5 through 8 for the other clusters.**

10 **When all server instances have the new application and are running, you can enable dynamic reconfiguration for all clusters again.**

Upgrading Incompatible Applications

If the new version of the application is incompatible with the old version, use the following procedure. For information on what makes applications compatible, see "Application Compatibility" on page 92. Also, you must upgrade incompatible application in two or more clusters. If you have only one cluster, create a "shadow cluster" for the upgrade, as described below.

When upgrading an incompatible application:

- Give the new version of the application a different name from the old version of the application. The steps below assume that the application is renamed.
- If the data schemas are incompatible, use different physical data sources after planning for data migration.
- Deploy the new version to a different cluster from the cluster where the old version is deployed.
- Set an appropriately long timeout for the cluster running the old application before you take it offline, because the requests for the application won't fail over to the new cluster. These user sessions will simply fail.

▼ To Upgrade an Incompatible Application by Creating a Second Cluster

1. **Save an old version of the application or back up the domain.**

   To back up the domain use the `asadmin backup-domain` command.

2. **Create a "shadow cluster" on the same or a different set of machines as the existing cluster. If you already have a second cluster, skip this step.**

   a. **Use the Admin Console to create the new cluster and reference the existing cluster's named configuration.**

      Customize the ports for the new instances on each machine to avoid conflict with existing active ports.

   b. **For all resources associated with the cluster, add a resource reference to the newly created cluster using** `asadmin create-resource-ref`**.**

   c. **Create a reference to all other applications deployed to the cluster (except the current redeployed application) from the newly created cluster using** `asadmin create-application-ref`**.**

   d. **Configure the cluster to be highly available using** `asadmin configure-ha-cluster`**.**

e. **Create reference to the newly-created cluster in the load balancer configuration file using** `asadmin create-http-lb-ref`.

3 **Give the new version of application a different name from the old version.**

4 **Deploy the new application with the new cluster as the target. Use a different context root or roots.**

5 **Enable the deployed new application for the clusters using** `asadmin enable-http-lb-application`.

6 **Start the new cluster while the other cluster is still running.**

The start causes the cluster to synchronize with the domain and be updated with the new application.

7 **Test the application on the new cluster to make sure it runs correctly.**

8 **Disable the old cluster from the load balancer using** `asadmin disable-http-lb-server`.

9 **Set a timeout for how long lingering sessions survive.**

10 **Enable the new cluster from the load balancer using** `asadmin enable-http-lb-server`.

11 **Export the load balancer configuration file using** `asadmin export-http-lb-config`.

12 **Copy the exported configuration file to the web server instance's configuration directory.**

For example, for Sun Java System Web Server, the location is *web-server-install-dir*/`https-`*host-name*/`config/loadbalancer.xml`. Dynamic reconfiguration must be enabled for the load balancer (by setting the `reloadinterval` in the load balancer configuration), so that the new load balancer configuration file is loaded automatically.

13 **After the timeout period expires or after all users of the old application have exited, stop the old cluster and delete the old application.**

Monitoring the HTTP Load Balancer Plug-in

- "Configuring Log Messages" on page 98
- "Types of Log Messages" on page 98
- "Enabling HTTP Load Balancer Logging" on page 99
- "Understanding Monitoring Messages" on page 101

Configuring Log Messages

The load balancer plug-in uses the web server's logging mechanism to write log messages. The default log level on the Application Server is set to the default logging level on Sun Java System Web Server (`INFO`), Apache Web Server (`WARN`) and Microsoft IIS `(INFO)`. The application server log levels, `FINE`, `FINER`, and `FINEST`, map to the `DEBUG` level on the web server.

These log messages are written to the web server log files, and are in the form of raw data that can be parsed using scripts, or imported into spreadsheets to calculate required metrics.

Types of Log Messages

The load balancer plug-in generates the following types of log messages:

- "Load Balancer Configurator Log Messages" on page 98
- "Request Dispatch and Runtime Log Messages" on page 99
- "Configurator Error Messages" on page 99

Load Balancer Configurator Log Messages

These messages will be logged when you are using idempotent URLs and error page settings.

An output for idempotent URL pattern configuration contains the following information:

- When the log level is set to FINE:

  ```
  CONFxxxx: IdempotentUrlPattern configured <url-pattern> <no-of-retries> for
  web-module : <web-module>
  ```
- When the log level is set to SEVERE:

  ```
  CONFxxxx: Duplicate entry of Idempotent URL element <url-pattern> for
  webModule <web-module> in loadbalancer.xml."
  ```
- When the log level is set to WARN:

  ```
  CONFxxxx: Invalid IdempotentUrlPatternData <url-pattern> for web-module
  <web-module>
  ```

  An output for error page URL configuration contains the following information (log level set to WARN):

  ```
  CONFxxxx: Invalid error-url for web-module <web-module>
  ```

Request Dispatch and Runtime Log Messages

These log messages are generated while a request is being load balanced and dispatched.

- An output for standard logging for each method start contains the following information (log level set to FINE):

  ```
  ROUTxxxx: Executing Router method <method_name>
  ```

- An output for router logs for each method start contains the following information (log level set to INFO):

  ```
  ROUTxxxx: Successfully Selected another ServerInstance for idempotent request
  <Request-URL>
  ```

- An output for runtime logs contains the following information (log level set to INFO):

  ```
  RNTMxxxx: Retrying Idempotent <GET/POST/HEAD> Request <Request-URL>
  ```

Configurator Error Messages

These errors appear if there are configuration problems, for example, if the custom error page referenced is missing.

- Log level set to INFO:

  ```
  ROUTxxxx: Non Idempotent Request <Request-URL> cannot be retried
  ```

  For example: `ROUTxxxx: Non Idempotent Request http://sun.com/addToDB?x=11&abc=2 cannot be retried`

- Log level set to FINE:

  ```
  RNTMxxxx: Invalid / Missing Custom error-url / page: <error-url> for
  web-module: <web-module>
  ```

  For example: `RNTMxxxx: Invalid / Missing Custom error-url / page: myerror1xyz for web-module: test`

Enabling HTTP Load Balancer Logging

The load balancer plug-in logs the following information:

- Request start/stop information for every request.
- Failed-over request information when the request fails over from an unhealthy instance to a healthy instance.
- List of unhealthy instances at the end of every health check cycle.

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

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

▼ To Turn on HTTP Load Balancer Logging

1. **Set the logging options in the web server. The procedure depends on the web server:**

   - **With Sun Java System Web Server**

     In the server's Admin console, go to the `Magnus Editor` tab and set the `Log Verbose` option to `On`.

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

   - **For Microsoft IIS, set the log level to `FINE` in the `sun-passthrough.properties` file.**

2. **Set the load balancer configuration's `monitor` option to true.**

   Use the `asadmin create-http-lb-config` command to set monitoring to true when you initially create the load balancer configuration, or use the `asadmin set` command to set it to true later. Monitoring is disabled by default.

Understanding Monitoring Messages

The format of the load balancer plug-in log messages is as follows.

- The start of an HTTP request contains the following information:

  ```
  RequestStart Sticky(New) <req-id> <time-stamp> <URL>
  ```

  The timestamp value is the number of milliseconds from January 1, 1970. For example:

  ```
  RequestStart New 123456 602983
  http://austen.sun.com/Webapps-simple/servlet/Example1
  ```

- The end of an HTTP request contains the `RequestExit` message, as follows:

  ```
  RequestExit Sticky(New) <req-id> <time-stamp> <URL> <listener-id>
  <response-time> Failure-<reason for error>(incase of a failure)
  ```

  For example:

  ```
  RequestExit New 123456 603001
  http://austen.sun.com/Webapps-simple/servlet/Example1 http://austen:2222 18
  ```

  Note – In the `RequestExit` message, *response-time* indicates the total request turnaround time in milliseconds, from the perspective of the load balancer plug-in.

- The list of unhealthy instances, as follows:

  ```
  UnhealthyInstances <cluster-id> <time-stamp> <listener-id>, <listener-id>...
  ```

  For example:

  ```
  UnhealthyInstances cluster1 701923 http://austen:2210, http://austen:3010
  ```

- A list of failed-over requests, as follows:

  ```
  FailedoverRequest <req-id> <time-stamp> <URL> <session-id>
  <failed-over-listener-id> <unhealthy-listener-id>
  ```

  For example:

  ```
  FailedoverRequest 239496 705623
  http://austen.sun.com/Apps/servlet/SessionTest 16dfdac3c7e80a40
  http://austen:4044 http://austen:4045
  ```

◆ ◆ ◆ CHAPTER 5

Using Application Server Clusters

This chapter describes how to use Application Server clusters. It contains the following sections:

- "Overview of Clusters" on page 103
- "Group Management Service" on page 103
- "Working with Clusters" on page 105

Overview of Clusters

A *cluster* is a named collection of server instances that share the same applications, resources, and configuration information. You can group server instances on different machines into one logical cluster and administer them as one unit. You can easily control the lifecycle of a multi-machine cluster with the DAS.

Clusters enable horizontal scalability, load balancing, and failover protection. By definition, all the instances in a cluster have the same resource and application configuration. When a server instance or a machine in a cluster fails, the load balancer detects the failure, redirects traffic from the failed instance to other instances in the cluster, and recovers the user session state. Since the same applications and resources are on all instances in the cluster, an instance can failover to any other instance in the cluster.

Group Management Service

The Group Management Service (GMS) is an infrastructure component that is enabled for the instances in a cluster. When GMS is enabled, if a clustered instance fails, the cluster and the Domain Administration Server are aware of the failure and can take action when failure occurs. Many features of Application Server depend upon GMS. For example, GMS is used by the IIOP failover, in-memory replication, transaction service, and timer service features.

If server instances in a cluster are located on different machines, ensure that the machines are on the same subnet.

Note – The GMS feature is not available in the developer profile. In the cluster profile and the enterprise profile, GMS is enabled by default.

GMS is a core service of the Shoal framework. For more information about Shoal, visit the Project Shoal home page (`https://shoal.dev.java.net/`).

▼ To Enable or Disable GMS for a Cluster

1. **In the tree component, select Clusters.**
2. **Click the name of the cluster.**
3. **Under General Information, ensure that the Heartbeat Enabled checkbox is checked or unchecked as required.**
   - **To enable GMS, ensure that the Heartbeat Enabled checkbox is checked.**
   - **To disable GMS, ensure that the Heartbeat Enabled checkbox is unchecked.**
4. **If you are enabling GMS and require different values for these defaults, change the default port and IP address for GMS.**
5. **Click Save.**

Configuring GMS

Configure GMS for your environment by changing the settings that determine how frequently GMS checks for failures. For example, you can change the timeout between failure detection attempts, the number of retries on a suspected failed member, or the timeout when checking for members of a cluster.

To configure monitoring in the Admin Console, go to Communications Application Server node –> Configuration –> Group Management Service.

The equivalent `asadmin` commands are `get` and `set`.

Working with Clusters

▼ To Create a Cluster

1. **In the tree component, select the Clusters node.**

2. **On the Clusters page, click New.**

   The Create Cluster page appears.

3. **In the Name field, type a name for the cluster.**

   The name must:

   - Consist only of uppercase and lowercase letters, numbers, underscores, hyphens, and periods (`.`)
   - Be unique across all node agent names, server instance names, cluster names, and configuration names
   - Not be `domain`

4. **In the Configuration field, choose a configuration from the drop-down list.**

   - **To create a cluster that does not use a shared configuration, choose `default-config`.**

     Leave the radio button labeled "Make a copy of the selected Configuration" selected. The copy of the default configuration will have the name *cluster_name*`-config`.

   - **To create a cluster that uses a shared configuration, choose the configuration from the drop-down list.**

     Select the radio button labeled "Reference the selected Configuration" to create a cluster that uses the specified existing shared configuration.

5 **Optionally, add server instances.**

You can also add server instances after the cluster is created.

Before you create server instances for the cluster, first create one or more node agents or node agent placeholders. See "To Create a Node Agent Placeholder" on page 131

To create server instances:

a. **In the Server Instances To Be Created area, click Add.**

b. **Type a name for the instance in the Instance Name field**

c. **Choose a node agent from the Node Agent drop-down list.**

6 **Click OK.**

7 **Click OK on the Cluster Created Successfully page that appears.**

More Information Equivalent asadmin command

```
create-cluster
```

See Also

- "To Configure a Cluster" on page 107
- "To Create Server Instances for a Cluster" on page 106
- "To Configure Applications for a Cluster" on page 109
- "To Configure Resources for a Cluster" on page 109
- "To Delete a Cluster" on page 110
- "To Upgrade Components Without Loss of Service" on page 111

For more details on how to administer clusters, server instances, and node agents, see "Deploying Node Agents" on page 123.

▼ To Create Server Instances for a Cluster

Before You Begin Before you can create server instances for a cluster, you must first create a node agent or node agent placeholder. See "To Create a Node Agent Placeholder" on page 131

1 **In the tree component, expand the Clusters node.**

2 **Select the node for the cluster.**

3 **Click the Instances tab to bring up the Clustered Server Instances page.**

4 **Click New to bring up the Create Clustered Server Instance page.**

5 **In the Name field, type a name for the server instance.**

6 **Choose a node agent from the Node Agents drop-down list.**

7 **Click OK.**

More Information Equivalent asadmin command

`create-instance`

See Also
- "What is a Node Agent?" on page 121
- "To Create a Cluster" on page 105
- "To Configure a Cluster" on page 107
- "To Configure Applications for a Cluster" on page 109
- "To Configure Resources for a Cluster" on page 109
- "To Delete a Cluster" on page 110
- "To Upgrade Components Without Loss of Service" on page 111
- "To Configure Server Instances in a Cluster" on page 108

▼ To Configure a Cluster

1 **In the tree component, expand the Clusters node.**

2 **Select the node for the cluster.**

On the General Information page, you can perform these tasks:

- Click Start Instances to start the clustered server instances.
- Click Stop Instances to stop the clustered server instances.
- Click Migrate EJB Timers to migrate the EJB timers from a stopped server instance to another server instance in the cluster.

More Information Equivalent asadmin command

`start-cluster`, `stop-cluster`, `migrate-timers`

See Also
- "To Create a Cluster" on page 105
- "To Create Server Instances for a Cluster" on page 106
- "To Configure Applications for a Cluster" on page 109
- "To Configure Resources for a Cluster" on page 109
- "To Delete a Cluster" on page 110
- "To Upgrade Components Without Loss of Service" on page 111

- "To Migrate EJB Timers" on page 110

▼ To Start, Stop, and Delete Clustered Instances

1. **In the tree component, expand the Clusters node.**
2. **Expand the node for the cluster that contains the server instance.**
3. **Click the Instances tab to display the Clustered Server Instances page.**

   On this page you can:

   - Select the checkbox for an instance and click Delete, Start, or Stop to perform the selected action on all the specified server instances.
   - Click the name of the instance to bring up the General Information page.

▼ To Configure Server Instances in a Cluster

1. **In the tree component, expand the Clusters node.**
2. **Expand the node for the cluster that contains the server instance.**
3. **Select the server instance node.**
4. **On the General Information page, you can:**
   - Click Start Instance to start the instance.
   - Click Stop Instance to stop a running instance.
   - Click JNDI Browsing to browse the JNDI tree for a running instance.
   - Click View Log Files to open the server log viewer.
   - Click Rotate Log File to rotate the log file for the instance. This action schedules the log file for rotation. The actual rotation takes place the next time an entry is written to the log file.
   - Click Recover Transactions to recover incomplete transactions.
   - Click the Properties tab to modify the port numbers for the instance.
   - Click the Monitor tab to change monitoring properties.

See Also

- "To Create a Cluster" on page 105
- "To Configure a Cluster" on page 107
- "To Create Server Instances for a Cluster" on page 106

- "To Configure Applications for a Cluster" on page 109
- "To Configure Resources for a Cluster" on page 109
- "To Delete a Cluster" on page 110
- "To Upgrade Components Without Loss of Service" on page 111
- "To configure how the Application Server recovers from transactions" in *Sun Java System Application Server 9.1 Administration Guide*

▼ To Configure Applications for a Cluster

1. **In the tree component, expand the Clusters node.**

2. **Select the node for the cluster.**

3. **Click the Applications tab to bring up the Applications page.**

   On this page, you can:

   - From the Deploy drop-down list, select a type of application to deploy. On the Deployment page that appears, specify the application.
   - From the Filter drop-down list, select a type of application to display in the list.
   - To edit an application, click the application name.
   - Select the checkbox next to an application and choose Enable or Disable to enable or disable the application for the cluster.

See Also

- "To Create a Cluster" on page 105
- "To Configure a Cluster" on page 107
- "To Create Server Instances for a Cluster" on page 106
- "To Configure Resources for a Cluster" on page 109
- "To Delete a Cluster" on page 110
- "To Upgrade Components Without Loss of Service" on page 111

▼ To Configure Resources for a Cluster

1. **In the tree component, expand the Clusters node.**

2. **Select the node for the cluster.**

3. **Click the Resources tab to bring up the Resources page.**

   On this page, you can:

- Create a new resource for the cluster: from the New drop-down list, select a type of resource to create. Make sure to specify the cluster as a target when you create the resource.
- Enable or Disable a resource globally: select the checkbox next to a resource and click Enable or Disalbe. This action does not remove the resource.
- Display only resources of a particular type: from the Filter drop-down list, select a type of resource to display in the list.
- Edit a resource: click the resource name.

See Also

- "To Create a Cluster" on page 105
- "To Configure a Cluster" on page 107
- "To Create Server Instances for a Cluster" on page 106
- "To Configure Applications for a Cluster" on page 109
- "To Delete a Cluster" on page 110

▼ To Delete a Cluster

1. **In the tree component, select the Clusters node.**
2. **On the Clusters page, select the checkbox next to the name of the cluster.**
3. **Click Delete.**

More Information

Equivalent asadmin command

```
delete-cluster
```

See Also

- "To Create a Cluster" on page 105
- "To Configure a Cluster" on page 107
- "To Create Server Instances for a Cluster" on page 106
- "To Configure Applications for a Cluster" on page 109
- "To Configure Resources for a Cluster" on page 109
- "To Upgrade Components Without Loss of Service" on page 111

▼ To Migrate EJB Timers

If a server instance stops running abnormally or unexpectedly, it can be necessary to move the EJB timers installed on that server instance to a running server instance in the cluster. To do so, perform these steps:

1. **In the tree component, expand the Clusters node.**

2 **Select the node for the cluster.**

3 **On the General Information page, click Migrate EJB Timers.**

4 **On the Migrate EJB Timers page:**

a. **From the Source drop-down list, choose the stopped server instance from which to migrate the timers.**

b. **(Optional) From the Destination drop-down list, choose the running server instance to which to migrate the timers.**

If you leave this field empty, a running server instance will be randomly chosen.

c. **Click OK.**

5 **Stop and restart the Destination server instance.**

If the source server instance is running or if the destination server instance is not running, Admin Console displays an error message.

More Information Equivalent asadmin command

```
migrate-timers
```

See Also

- "To Configure a Cluster" on page 107
- Admin Console online help for configuring settings for the EJB timer service

▼ To Upgrade Components Without Loss of Service

You can use the load balancer and multiple clusters to upgrade components within the Communications Application Server without any loss of service. A component can, for example, be a JVM, the Communications Application Server, or a web application.

This approach is not possible if:

- You perform an application upgrade that involves a change to the application database schema.

Caution – Upgrade all server instances in a cluster together. Otherwise, there is a risk of version mismatch caused by a session failing over from one instance to another where the instances have different versions of components running.

1 **Stop one of the clusters using the Stop Cluster button on the General Information page for the cluster.**

2 **Upgrade the component in that cluster.**

3 **Start the cluster using the Start Cluster button on the General Information page for the cluster.**

4 **Repeat the process with the other clusters, one by one.**

Because sessions in one cluster will never fail over to sessions in another cluster, there is no risk of version mismatch caused by a session's failing over from a server instance that is running one version of the component to another server instance (in a different cluster) that is running a different version of the component. A cluster in this way acts as a safe boundary for session failover for the server instances within it.

See Also

- "To Create a Cluster" on page 105
- "To Configure a Cluster" on page 107
- "To Create Server Instances for a Cluster" on page 106
- "To Configure Applications for a Cluster" on page 109
- "To Configure Resources for a Cluster" on page 109
- "To Delete a Cluster" on page 110

CHAPTER 6

Managing Configurations

This chapter describes adding, changing, and using named server configurations in Communications Application Server. It contains the following sections:

Using Configurations

Configurations

A configuration is a set of server configuration information, including settings for things such as HTTP/SIP listeners, ORB/IIOP listeners, JMS brokers, the EJB container, security, logging, and monitoring. Applications and resources are not defined in named configurations.

Configurations exist in an administrative domain. Multiple server instances or clusters in the domain can reference the same configuration, or they can have separate configurations.

For clusters, all server instances in the cluster inherit the cluster's configuration so that a homogenous environment is assured in a cluster's instances.

Because a configuration contains so many required settings, create a new configuration by copying an existing named configuration. The newly-created configuration is identical to the configuration you copied until you change its configuration settings.

There are three ways in which clusters or instances use configurations:

- **Stand-alone:** A stand-alone server instance or cluster doesn't share its configuration with another server instance or cluster; that is, no other server instance or cluster references the named configuration. You create a stand-alone instance or cluster by copying and renaming an existing configuration.
- **Shared:** A shared server instance or cluster shares a configuration with another server instance or cluster; that is, multiple instances or clusters reference the same named configuration. You create a shared server instance or cluster by referencing (not copying) an existing configuration.
- **Clustered:** A clustered server instance inherits the cluster's configuration.

  See Also:

- "The default-config Configuration" on page 114
- "Configurations Created when Creating Instances or Clusters" on page 114
- "Unique Port Numbers and Configurations" on page 115
- "To Create a Named Configuration" on page 116
- "Editing a Named Configuration's Properties" on page 116

The default-config Configuration

The `default-config` configuration is a special configuration that acts as a template for creating stand-alone server instance or stand-alone cluster configurations. Clusters and individual server instances cannot refer to `default-config`; it can only be copied to create new configurations. Edit the default configuration to ensure that new configurations copied from it have the correct initial settings.

For more information, see:

- "Configurations Created when Creating Instances or Clusters" on page 114
- "Configurations" on page 113
- "To Create a Named Configuration" on page 116
- "Editing a Named Configuration's Properties" on page 116
- "To Edit Port Numbers for Instances Referencing a Configuration" on page 118

Configurations Created when Creating Instances or Clusters

When creating a new server instance or a new cluster, either:

- Reference an existing configuration. No new configuration is added.
- Make a copy of an existing configuration. A new configuration is added when the server instance or cluster is added.

By default, new clusters or instances are created with configurations copied from the `default-config` configuration. To copy from a different configuration, specify it when creating a new instance or cluster.

For a server instance, the new configuration is named *instance_name*`-config` . For a cluster, the new configuration is named *cluster-name* `-config`.

For more information, see:

- "The default-config Configuration" on page 114
- "Configurations" on page 113
- "To Create a Named Configuration" on page 116
- "Editing a Named Configuration's Properties" on page 116

Clustered Configuration Synchronization

When you create a clustered configuration, Communications Application Server creates a cluster configuration directory on the domain administration server at *domain-root*`/`*domain-dir*`/config/cluster-config`. This directory is used to synchronize configurations for all instances in the cluster.

Unique Port Numbers and Configurations

If multiple instances on the same host machine reference the same configuration, each instance must listen on a unique port number. For example, if two server instances reference a named configuration with an HTTP listener on port 80, a port conflict prevents one of the server instances from starting. Change the properties that define the port numbers on which individual server instances listen so that unique ports are used.

The following principles apply to port numbers:

- Port numbers for individual server instances are initially inherited from the configuration.
- If the port is already in use when you create a server instance, override the inherited default value at the instance level to prevent port conflicts.
- Assume an instance is sharing a configuration. The configuration has port number n. If you create a new instance on the machine using the same configuration, the new instance is assigned port number *n*+1, if it is available. If it is not available, the next available port after *n*+1 is chosen.
- If you change the port number of the configuration, a server instance inheriting that port number automatically inherits the changed port number.
- If you change an instance's port number and you subsequently change the configuration's port number, the instance's port number remains unchanged.

  For more information, see:

- "To Edit Port Numbers for Instances Referencing a Configuration" on page 118

- "Editing a Named Configuration's Properties" on page 116
- "Configurations" on page 113

Working with Named Configurations

- "To Create a Named Configuration" on page 116
- "Editing a Named Configuration's Properties" on page 116
- "To Edit Port Numbers for Instances Referencing a Configuration" on page 118
- "To view a Named Configuration's Targets" on page 118
- "To Delete a Named Configuration" on page 119

▼ To Create a Named Configuration

1. **In the tree component, select the Configurations node.**
2. **On the Configurations page, click New.**
3. **On the Create Configurations page, enter a unique name for the configuration.**
4. **Select a configuration to copy.**

   The configuration `default-config` is the default configuration used when creating stand-alone server instance or stand-alone cluster.

More Information Equivalent asadmin command

```
copy-config
```

See Also

- "Configurations" on page 113
- "The default-config Configuration" on page 114
- "Editing a Named Configuration's Properties" on page 116
- "To Edit Port Numbers for Instances Referencing a Configuration" on page 118
- "To view a Named Configuration's Targets" on page 118
- "To Delete a Named Configuration" on page 119

Editing a Named Configuration's Properties

Remark 6–1 **Properties for SIP listeners?**

The following table describes the properties predefined for a configuration.

The predefined properties are port numbers. Valid port values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges. If more than one server instance exists on a system, the port numbers must be unique.

Property Name	Description
HTTP_LISTENER_PORT	Port number for http-listener-1.
HTTP_SSL_LISTENER_PORT	Port number for http-listener-2.
IIOP_SSL_LISTENER_PORT	ORB listener port for IIOP connections on which IIOP listener `SSL` listens.
IIOP_LISTENER_PORT	ORB listener port for IIOP connections on which `orb-listener-1` listens.
JMX_SYSTEM_CONNECTOR_PORT	Port number on which the JMX connector listens.
IIOP_SSL_MUTUALAUTH_PORT	ORB listener port for IIOP connections on which the IIOP listener `SSL_MUTUALAUTH` listens.

▼ To Edit a Named Configuration's Properties

1. **In the tree component, expand the Configurations node.**

2. **Select the node for a named configuration.**

3. **Select System Properties.**

4. **On the Configuration System Properties page, choose whether to enable dynamic reconfiguration.**

   If enabled, changes to the configuration are applied to the server instances without requiring a server restart.

5. **Add, delete, or modify properties as desired.**

6. **To edit the current values of a property for all instances associated with the configuration, click Instance Values.**

More Information

Equivalent asadmin command

```
set
```

See Also

- "Configurations" on page 113
- "To Create a Named Configuration" on page 116
- "To view a Named Configuration's Targets" on page 118

- "To Delete a Named Configuration" on page 119

▼ To Edit Port Numbers for Instances Referencing a Configuration

Each instance referencing a named configuration initially inherits its port numbers from that configuration. Since port numbers must be unique on the system, you might need to override the inherited port numbers.

1 **In the tree component, expand the Configurations node.**

2 **Select the node for a named configuration.**

3 **Select System Properties.**

The Admin Console displays the Configuration System Properties page.

4 **Click Instance Values next to the instance variable you want to edit.**

For example, if you click Instance Values next to the HTTP-LISTENER-PORT instance variable, you see the value of HTTP-LISTENER-PORT for every server instance that references that configuration.

5 **Change the values as desired and click Save.**

More Information Equivalent asadmin command

```
set
```

See Also

- "Unique Port Numbers and Configurations" on page 115
- "Configurations" on page 113
- "Editing a Named Configuration's Properties" on page 116

▼ To view a Named Configuration's Targets

The Configuration System Properties page displays a list of all targets using the configuration. For a cluster configuration, the targets are clusters. For an instance configuration, the targets are instances.

1 **In the tree component, expand the Configurations node.**

2 **Select a node for the named configuration.**

3 **Select System Properties.**

The Admin Console displays the Configuration System Properties page.

See Also

- "Unique Port Numbers and Configurations" on page 115
- "Configurations" on page 113
- "To Create a Named Configuration" on page 116
- "Editing a Named Configuration's Properties" on page 116
- "To Delete a Named Configuration" on page 119

▼ To Delete a Named Configuration

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

2 **On the Configurations page, select the checkbox for the named configuration to delete.**

You cannot delete the `default-config` configuration.

3 **Click Delete.**

More Information

Equivalent asadmin command

```
delete-config
```

See Also

- "Configurations" on page 113
- "To Create a Named Configuration" on page 116
- "Editing a Named Configuration's Properties" on page 116
- "To view a Named Configuration's Targets" on page 118

CHAPTER 7

Configuring Node Agents

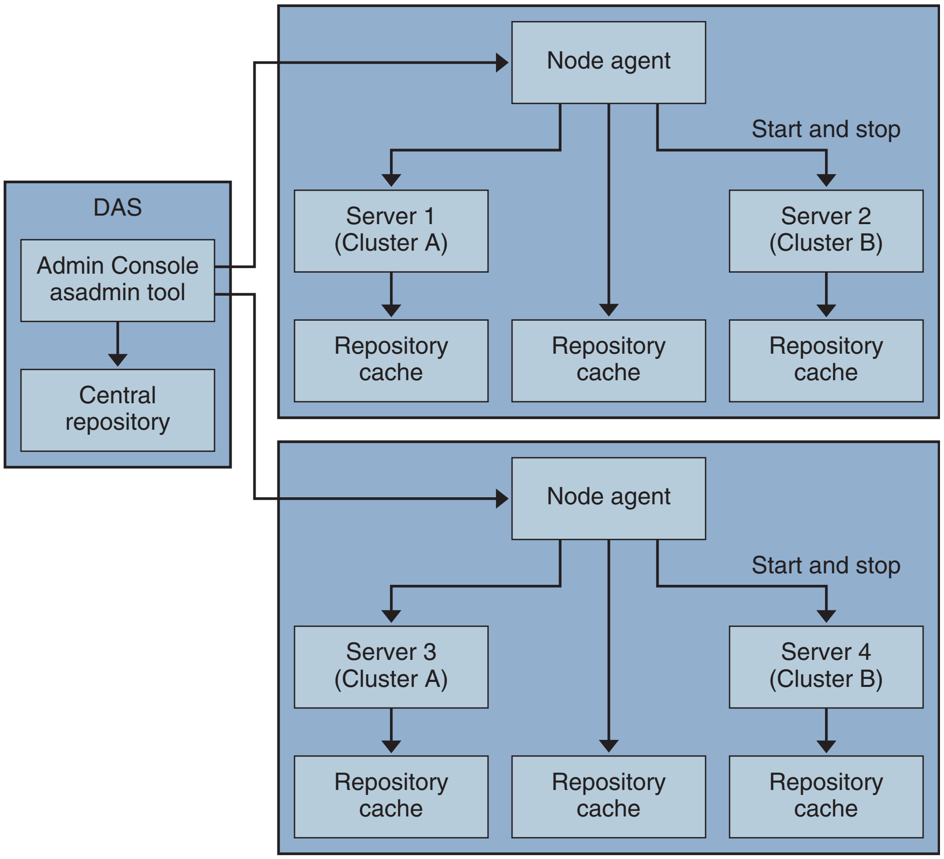
This chapter describes the node agents in Communications Application Server. It contains the following sections:

What is a Node Agent?

A node agent is a lightweight process that is required on every machine that hosts server instances, including the machine that hosts the Domain Administration Server (DAS). The node agent:

- Starts, stops, creates and deletes server instances as instructed by the Domain Administration Server.
- Restarts failed server instances.
- Provides a view of the log files of failed servers.
- Synchronizes each server instance's local configuration repository with the Domain Administration Server's central repository. Each local repository contains only the information pertinent to that server instance or node agent.

  The following figure illustrates the overall node agent architecture:



When you install the Application Server, a node agent is created by default with the host name of the machine. This node agent must be manually started on the local machine before it runs.

You can create and delete server instances even if the node agent is not running. However, the node agent must be running before you use it to start and stop server instances.

A node agent services a single domain. If a machine hosts instances running in multiple domains, it must run multiple node agents.

Server Instance Behavior After Node Agent Failure

A node agent might be stopped unexpectedly, for example, by a software failure or other error. In this situation, any server instances that the node agent was managing are no longer managed. However, such server instances continue to run and remain accessible by the DAS. Information about the server instances can still be obtained through Communications Application Server administrative interfaces, and applications that are deployed on the server instances can still be accessed.

If the node agent is restarted, the server instances that are not managed remain unmamaged. The node agent does *not* resume the management of these server instances. If an unmanaged server instance is stopped unexpectedly, for example, by a software failure or other error, the node agent cannot restart the server instance.

If an unmanaged server instance must continue to run, you cannot resume the management of the server instance by a node agent. The only way to resume the management of an unmanaged server instance is to stop and restart the server instance after the node agent is restarted.

Deploying Node Agents

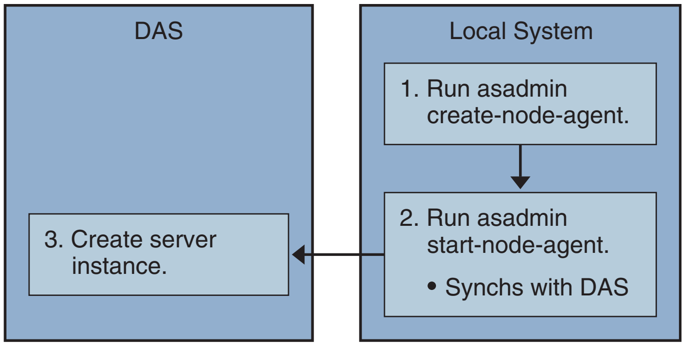
You configure and deploy node agents in two ways:

- *Online deployment*, when you know your topology and already have the hardware for your domain.
- *Offline deployment*, when you are configuring domains and server instances before setting up the full environment

▼ To Deploy Node Agents Online

Use online deployment if you already know the domain topology and have the hardware for your domain.

The following figure summarizes the online deployment of node agents:


DAS
Local System
1. Run asadmin create-node-agent.
2. Run asadmin start-node-agent.
• Synchs with DAS
3. Create server instance.

Before You Begin Install and start the Domain Administration Server. Once the Domain Administration Server is up and running, begin either online or offline deployment.

1 **Install a node agent on every machine that will host a server instance.**

Use the installer or the `asadmin` `create-node-agent` command . If a machine requires more than one node agent, use the `asadmin` `create-node-agent` command to create them.

See "Creating a Node Agent" on page 132 for more information.

2 **Start the node agents using the `asadmin` `start-node-agent` command .**

When started, a node agent communicates with the Domain Administration Server (DAS). When it reaches the DAS, a configuration for the node agent is created on the DAS. Once the configuration exists, the node agent is viewable in the Admin Console.

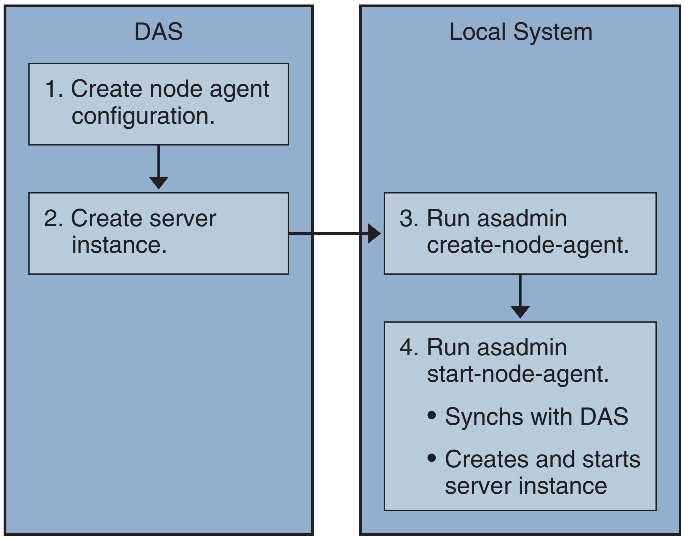
See "Starting a Node Agent" on page 134 for more information.

3 **Configure the domain: create server instances, create clusters, and deploying applications.**

▼ To Deploy Node Agents Offline

Use offline deployment to deploy node agents in the domain before configuring the individual local machines.

The following figure summarizes the offline deployment.


DAS
1. Create node agent configuration.
2. Create server instance.
Local System
3. Run asadmin create-node-agent.
4. Run asadmin start-node-agent.
• Synchs with DAS
• Creates and starts server instance

Before You Begin Install and start the Domain Administration Server. Once the Domain Administration Server is up and running, begin either online or offline deployment.

1 **Create placeholder node agents in the Domain Administration Server.**

See "To Create a Node Agent Placeholder" on page 131 for more information.

2 **Create server instances and clusters, and deploy applications.**

When creating a server instance, make sure to assign port numbers that are not already in use. Because the configuration is being done offline, the domain cannot check for port conflicts at creation time.

3 **Install a node agent on every machine that will host a server instance.**

Use the installer or the `asadmin` `create-node-agent` command . The node agents must have the same names as the placeholder node agents previously created.

See "Creating a Node Agent" on page 132 for more information.

4 **Start the node agents using the `asadmin` `start-node-agent` command .**

When a node agent is started, it binds to the Domain Administration Server and creates any server instances previously associated with the node agent.

See "Starting a Node Agent" on page 134 for more information.

Synchronizing Node Agents and the Domain Administration Server

Because configuration data is stored in the Domain Administration Server's repository (the central repository) and also cached on the node agent's local machine, the two must be synchronized. The synchronization of cache is always done on a explicit user action through the administration tools.

This section contains the following topics:

- "Node Agent Synchronization" on page 125
- "Server Instance Synchronization" on page 126
- "Synchronizing Library Files" on page 128
- "Unique Settings and Configuration Management" on page 128
- "Synchronizing Large Applications" on page 129

Node Agent Synchronization

When a node agent is started for the first time, it sends a request to the Domain Administration Server (DAS) for the latest information in the central repository. Once it successfully contacts the DAS and gets configuration information, the node agent is bound to that DAS.

Note – By default, the `asadmin start-node-agent` command automatically starts the remote server instances without synchronizing with DAS. If you are starting a remote server instance that is synchronized with the central repository managed by DAS, specify the `--startinstances=false` option of the `asadmin start-node-agent` command. Then use the `asadmin start-instance` command to start the remote server instance.

If you created a placeholder node agent on the DAS, when the node agent is started for the first time it gets its configuration from the central repository of the DAS. During its initial start-up, if the node agent is unable to reach the DAS because the DAS is not running, the node agent stops and remains unbound.

If changes are made in the domain to the node agent's configuration, they are automatically communicated to the node agent on the local machine while the node agent is running.

If you delete a node agent configuration on the DAS, the next time the node agent synchronizes, it stops and marks itself as awaiting deletion. Manually delete it using the local `asadmin delete-node-agent` command.

Server Instance Synchronization

If you explicitly start a server instance with the Admin Console or `asadmin` tool, the server instance is synchronized with the central repository. If this synchronization fails, the server instance doesn't start.

If a node agent starts a server instance without an explicit request through the Admin Console or the `asadmin` tool, the repository cache for the server instance is not synchronized. The server instance runs with the configuration as stored in its cache. You must not add or remove files in the remote server instance's cache.

The remote server instance's configuration are treated as cache (all files under `nodeagents/`*na1*/*server1*) and owned by Application Server. In extreme cases, if user removes all files of a remote server instance and restarts the node agent, the remote server instance (for example, *server1*) will be recreated and all necessary files will be synchronized.

The following files and directories are kept synchronized by the Application Server.

TABLE 7–1 Files and directories synchronized among remote server instances

File or directory	Description
`applications`	All deployed applications. The parts of this directory (and sub directories) synchronized depend on the applications referred to from the server instance. The Node agent does not synchronize any of the applications because it does not reference any application.
`config`	Contains configuration files for the entire domain. All the files in this directory are synchronized except runtime temporary files, such as, `admch`, `admsn`, `secure.seed`, `.timestamp`, and `__timer_service_shutdown__.dat`.
`config/`*config_name*	Directory to store files to be shared by all instances using config named *config_name*. There will be one such directory for every config defined in domain.xml. All the files in this directory are synchronized to the server instances that are using the *config_name*.
`config/`*config_name*`/ lib/ext`	Folder where Java extension classes (as zip or jar archives) can be dropped. This is used by applications deployed to server instances using config named *config_name*. These jar files are loaded using Java extension mechanism.
`docroot`	The HTTP document root. In out of the box configuration, all server instances in the domain use the same docroot. The docroot property of the virtual server needs to be configured to make the server instances use a different docroot.
`generated`	Generated files for Java EE applications and modules, for example, EJB stubs, compiled JSP classes, and security policy files. This directory is synchronized along with applications directory. Therefore, only the directories corresponding to applications referenced by a server instance are synchronized.
`lib`, `lib/classes`	Folder where common Java class files or jar and zip archives used by applications deployed to entire domain can be dropped. These classes are loaded using Application Server's class loader. The load order in class loader is: `lib/classes`, `lib/*.jar`, `lib/*.zip`.
`lib/ext`	Folder where Java extension classes (as zip or jar archives) used by applications deployed to entire domain can be dropped. These jar files are loaded using Java extension mechanism.
`lib/applibs`	Place dependent jars under `domains/<`*domain_name*`>lib/applibs` and specify a relative path to the jar file through the `libraries` option. For example, `asadmin deploy --libraries commons-coll.jar,X1.jar foo.ear`
`java-web-start`	The parts of this directory (and sub directories) are synchronized depending on the applications referred to from the server instance.

Synchronizing Library Files

The `--libraries` deploy time attribute for an application can be used to specify runtime dependencies of an application. When a relative path is specified, (only the jar name), Application Server attempts to find the specified library in *domain-dir*`/lib/applibs`.

To make a library available to the whole domain, you could place the JAR file in *domain-dir*`/lib` or *domain-dir*`/lib/classes`. (For more information, see "Using the Common Class Loader" in *Sun Java System Application Server 9.1 Developer's Guide*.) This is usually the case for JDBC drivers and other utility libraries that are shared by all applications in the domain.

For cluster-wide or stand alone server wide use, copy the jars into the *domain-dir*`/domain1/config/`*xyz-config*`/lib` directory. Next, add the jars in `classpath-suffix` or `classpath-prefix` element of *xyz-config*. This will synchronize the jars for all server instances using *xyz-config*.

In summary:

- `domains/domain1/lib` - domain wide scope, common class loader, adds the jars automatically.
- `domains/domain1/config/cluster1`, `config/lib` - config wide, update `classpath-prefix` or `classpath-suffix`.
- `domains/domain1/lib/applibs` - application scope, added to application class loader automatically
- `domains/domain1/config/cluster1`, `config/lib/ext` - adds to `http://java.sun.com/j2se/1.5.0/docs/guide/extensions/extensions.html` automatically.

Unique Settings and Configuration Management

Configuration files (under `domains/`*domain1*`/config` are synchronized across the domain. If you want to customize a `server.policy` file for a *server1*`-config` used by a stand alone server instance (*server1*), place the modified `server.policy` file under `domains/domain1/config/server1-config` directory.

This modified `server.policy` file will only be synchronized for the stand alone server instance, *server1*. You should remember to update the `jvm-option`. For example:

```
<java-config>
...
<jvm-options>-Djava.security.policy=${com.sun.aas.instanceRoot}/config
/server1-config/server.policy</jvm-options>
</java-config>
```

Synchronizing Large Applications

When your environment contains large applications to synchronize or available memory is constrained, you can adjust the JVM options to limit memory usage. This adjustment reduces the possibility of receiving out of memory errors. The instance synchronization JVM uses default settings, but you can configure JVM options to change them.

Set the JVM options using the `INSTANCE-SYNC-JVM-OPTIONS` property. The command to set the property is:

```
asadmin set
domain.node-agent.node_agent_name.property.INSTANCE-SYNC-JVM-OPTIONS="JVM_options"
```

For example:

```
asadmin set
domain.node-agent.node0.property.INSTANCE-SYNC-JVM-OPTIONS="-Xmx32m -Xss2m"
```

In this example, the node agent is `node0` and the JVM options are `-Xmx32m -Xss2m`.

For more information, see `http://java.sun.com/docs/hotspot/VMOptions.html`.

Note – Restart the node agent after changing the INSTANCE-SYNC-JVM-OPTIONS property, because the node agent is not automatically synchronized when a property is added or changed in its configuration.

Using the doNotRemoveList Flag

If your application requires to store and read files in the directories (applications, generated, docroot, config, lib, java-web-start) that are synchronized by the Application Server, use the `doNotRemoveList` flag. This attribute takes a coma-separated list of files or directories. Your application dependent files are not removed during server startup, even if they do not exist in the central repository managed by DAS. If the same file exists in the central repository, they will be over written during synchronization.

Use the `INSTANCE-SYNC-JVM-OPTIONS` property to pass in the doNotRemoveList attribute.

For example:

```
<node-agent name="na1" ...>

...

<property name="INSTANCE-SYNC-JVM-OPTIONS"
value="-Dcom.sun.appserv.doNotRemoveList=applications/j2ee-modules
/<webapp_context>/logs,generated/mylogdir"/>
```

```
</node–agent>
```

Viewing Node Agent Logs

Each node agent has its own log file. If you experience problems with a node agent, see the log file at:

node_agent_dir /*node_agent_name*/`agent/logs/server.log` .

Sometimes the node agent log instructs you to look at a server's log to get a detailed message about a problem.

The server logs are located at:

node_agent_dir/*node_agent_name*/ *server_name*/`logs/server.log`

The default location for *node_agent_dir* is *install_dir*/`nodeagents`.

Working with Node Agents

- "How to Perform Node Agent Tasks" on page 130
- "Node Agent Placeholders" on page 131
- "To Create a Node Agent Placeholder" on page 131
- "Creating a Node Agent" on page 132
- "Starting a Node Agent" on page 134
- "Stopping a Node Agent" on page 135
- "Deleting a Node Agent" on page 135
- "To View General Node Agent Information" on page 135
- "To Delete a Node Agent Configuration" on page 136
- "To Edit a Node Agent Configuration" on page 137
- "To Edit a Node Agent Realm" on page 137
- "To Edit the Node Agent's Listener for JMX" on page 138

How to Perform Node Agent Tasks

Some node agent tasks require you to use the `asadmin`tool locally on the system where the node agent runs. Other tasks you can perform remotely using either the Admin Console or `asadmin`.

The following table summarizes the tasks and where to run them:

TABLE 7–2 How To Perform Node Agent Tasks

Task	Admin Console	asadmin Command
Create node agent placeholder on Domain Administration Server	Create Node Agent placeholder page	create-node-agent-config
Create node agent	Not available	create-node-agent
Start node agent	Not available	start-node-agent
Stop node agent	Not available	stop-node agent
Delete node agent configuration from Domain Administration Server	Node Agents page	delete-node-agent-config
Delete node agent from local machine	Not available	delete-node-agent
Edit node agent configuration	Node Agents pages	set
List node agents	Node Agents page	list-node-agents

Node Agent Placeholders

You can create and delete server instances without an existing node agent using a *node agent placeholder*. The node agent placeholder is created on the Domain Administration Server (DAS) before the node agent itself is created on the node agent's local system.

For information on creating a node agent placeholder, see "To Create a Node Agent Placeholder" on page 131

Note – Once you've created a placeholder node agent, use it to create instances in the domain. However, before starting the instances you must create and start the actual node agent locally on the machine where the instances will reside using the `asadmin` command. See "Creating a Node Agent" on page 132 and "Starting a Node Agent" on page 134

▼ To Create a Node Agent Placeholder

A node agent is a local watchdog for server instances that are running on a remote machine. Therefore, a node agent must be created on the machine that is hosting the server instances. As a result of this requirement, you can use the Admin Console to create only a placeholder for a node agent. This placeholder is a node agent configuration for which a node agent does not yet exist.

After creating a placeholder, use the `asadmin` command `create-node-agent` on the machine hosting the node agent to complete the creation. For more information, see "Creating a Node Agent" on page 132.

For a list of the steps involved in creating and using node agents, see "Deploying Node Agents" on page 123.

1 **In the tree component, select the Node Agents node.**

2 **On the Node Agents page, click New.**

3 **On the Current Node Agent Placeholder page, enter a name for the new node agent.**

The name must be unique across all node agent names, server instance names, cluster names, and configuration names in the domain.

4 **Click OK.**

The placeholder for your new node agent is listed on the Node Agents page.

More Information

Equivalent asadmin command

`create-node-agent-config`

Creating a Node Agent

To create a node agent, run the `asadmin` command `create-node-agent` locally on the machine on which the node agent runs.

The default name for a node agent is the host name on which the node agent is created.

If you've already created a node agent placeholder, use the same name as the node agent placeholder to create the associated node agent. If you have not created a node agent placeholder, and the DAS is up and reachable, the `create-node-agent` command also creates a node agent configuration (placeholder) on the DAS.

For a complete description of the command syntax, see the online help for the command.

The DAS and a node agent might be configured to communicate securely. In this situation, when the node agent is started, it must validate the certificate that the DAS sends to the node agent. To validate the certificate, the node agent looks up the certificate in the node agent's local truststore , which is protected by a master password. To enable the node agent to be started without prompting the user for a password, save the node agent's master password to a file when you create the node agent. If you do not save the node agent's master password to a file, the user is prompted for the master password whenever the user starts the node agent.

Note – In some situations you must specify the name of a host that can be reached through DNS. For more information, see "To Create a Node Agent for a DNS-Reachable Host" on page 133.

▼ To Create a Node Agent

● **Type the following command:**

`asadmin create-node-agent --host` *das-host* **`--port`** *port-no* **`--user`** *das-user* `[`**`--savemasterpassword=true`**`]` *nodeagent*

To enable the node agent to be started without prompting the user for a password, save the node agent's master password to a file. To save the node agent's master password to a file, set the `--savemasterpassword` option to `true` in the command to create the node agent.

If you set `--savemasterpassword` to `true`, you are prompted for the master password. Otherwise, you are *not* prompted for a password.

`--host` *das-host*	Specifies the name of the host where the Domain Administration Server (DAS) is running.
`-port` *port-no*	Specifies the HTTP/HTTPS port number for administering the domain.
`--user` *das-user*	Specifies the DAS user.
nodeagent	Specifies the name of the node agent that you are creating. This name must be unique in the domain.

Example 7–1 Creating a Node Agent

```
asadmin create-node-agent --host myhost --port 4848 --user admin nodeagent1
```

This command creates a node agent that is named `nodeagent1`. The DAS with which the node agent communicates is running on the machine `myhost`. The HTTP port for administering the agent's domain is 4848. The name of the DAS user is `admin`.

▼ To Create a Node Agent for a DNS-Reachable Host

The host where the DAS is running must be reachable through DNS in the following situations:

- Domains cross subnet boundaries, that is, the node agent and the DAS are in different domains, for example, `sun.com` and `java.com`.
- A DHCP machine is being used whose host name is not registered in DNS.

1 **In the `create-domain` command to create the domain, specify the `--domainproperties` `domain.hostName=`*das-host-name* option.**

das-host-name is the name of the machine where the DAS is running.

2 **In the `create-node-agent` command to create the node agent, specify the following options:**

- `--host` *das-host-name*, where *das-host-name* is the DAS host name that you specified in Step 1. This option corresponds to the `agent.das.host` property in the file *as-install*`/nodeagents/`*nodeagentname*`/agent/config/das.properties`.
- `--agentproperties remoteclientaddress=`*node-agent-host-name*, where *node-agent-host-name* is the host name that the DAS uses to connect to the node agent. This option corresponds to the `agent.client.host` property in the file *as-install*`/nodeagents/`*nodeagentname*`/agent/config/nodeagent.properties`.

More Information

Specifying the Host by Updating the `hosts` File

Another solution is to update the `hosts` hostname/IP resolution file specific to the platform so the hostname resolves to the correct IP address. However, when reconnecting using DHCP you might get assigned a different IP address. In that case, you must update the host resolution files on each server.

Starting a Node Agent

Before a node agent can manage server instances, it must be running. Start a node agent by running the `asadmin` command `start-node-agent` locally on the system where the node agent resides.

For a complete description of the command syntax, see the online help for the command.

For example:

```
asadmin start-node-agent --user admin --startinstances=false nodeagent1
```

where `admin` is your administration user, and `nodeagent1` is the node agent being started.

By default, the cache repositories of node agent instances are not synchronized from the central repository when a node agent is restarted. To forcibly synchronize the instances' cache repositories with central repository, set the `--syncinstances` option to `true` in the `asadmin start-node-agent` command.

Note – if you set the `--syncinstances` option to `true`, the repositories of *all* instances are synchronized when the node agent is restarted.

After restarting a node agent, use the `asadmin start-instance` command to start the server instance.

Stopping a Node Agent

Run the asadmin command stop-node-agent on the system where the node agent resides to stop a running node agent. The stop-node-agent command stops all server instances that the node agent manages.

For a complete description of the command syntax, see the online help for the command.

For example:

```
asadmin stop-node-agent nodeagent1
```

where nodeagent1 is the name of the node agent.

Deleting a Node Agent

Before deleting a node agent, the node agent must be stopped. You can also delete a node agent if it has never been started, or never successfully able to contact the Domain Administration Server (that is, if it is still unbound).

Run the asadmin command delete-node-agent on the system where the node agent resides to delete the node agent files.

For a complete description of the command syntax, see the online help for the command.

For example:

```
asadmin delete-node-agent nodeagent1
```

where nodeagent1 is your node agent.

When deleting a node agent, you must also delete its configuration from the Domain Administration Server using either the Admin Console or the asadmin delete-node-agent-config command.

▼ To View General Node Agent Information

1 **In the tree component, select the Node Agents node.**

2 **Click the name of a node agent.**

If a node agent already exists but does not appear here, start the node agent on the node agent's host machine using asadmin start-node-agent. See "Starting a Node Agent" on page 134

3 **Check the node agent's host name.**

If the host name is Unknown Host, then the node agent has not made initial contact with the Domain Administration Server (DAS).

4 **Check the node agent status.**

The status can be:

- **Running**: The node agent has been properly created and is currently running.
- **Not Running**: Either the node agent has been created on the local machine, but never started or the node agent was started but has been stopped.
- **Waiting for Rendezvous**: The node agent is a placeholder that has never been created on the local machine.

See "Creating a Node Agent" on page 132 and "Starting a Node Agent" on page 134.

5 **Choose whether to start instances on start up.**

Select Yes to start server instances associated with the node agent automatically when the node agent is started. Select No to start the instances manually.

6 **Determine whether the node agent has made contact with the Domain Administration Server.**

If the node agent has never made contact with the Domain Administration Server, it has never been successfully started.

7 **Manage server instances associated with the node agent.**

If the node agent is running, start or stop an instance by clicking the checkbox next to the instance name and clicking Start or Stop.

▼ To Delete a Node Agent Configuration

Through the Admin Console you can only delete the node agent configuration from the domain. You cannot delete the actual node agent. To delete the node agent itself, run the `asadmin` command `delete-node-agent` on the node agent's local machine. For more information, see "Deleting a Node Agent" on page 135.

Before deleting the node agent configuration, the node agent must be stopped and it must not have any associated instances. To stop a node agent, use the `asadmin` command `stop-node-agent`. See "Stopping a Node Agent" on page 135 for more information.

1 **In the tree component, select the Node Agents node.**

2 **On the Node Agents page, select the checkbox next to the node agent to be deleted.**

3 **Click delete.**

More Information Equivalent asadmin command

```
delete-node-agent-config
```

▼ To Edit a Node Agent Configuration

1 **In the tree component, expand the Node Agents node.**

2 **Select the node agent configuration to edit.**

3 **Check Start Instances on Startup to start the agent's server instances when the agent is started.**

You can also manually start and stop instances from this page.

If this configuration is for a placeholder node agent, when you create the actual node agent using `asadmin create-node-agent` , it picks up this configuration. For information on creating a node agent, see "Creating a Node Agent" on page 132.

If this configuration is for an existing node agent, the node agent configuration information is synchronized automatically.

▼ To Edit a Node Agent Realm

You must set an authentication realm for users connecting to the node agent. Only administration users should have access to the node agent.

1 **In the tree component, expand the Node Agents node.**

2 **Select the node agent configuration to edit.**

3 **Click the Auth Realm tab.**

4 **On the Node Agents Edit Realm page, enter a realm.**

The default is `admin-realm`, created when you create the node agent. To use a different realm, replace the realms in *all* the components controlled by the domain or the components won't communicate properly.

5 **In the Class Name field, specify the Java class that implements the realm.**

6 **Add any required properties.**

Authentication realms require provider-specific properties, which vary depending on what a particular implementation needs.

▼ To Edit the Node Agent's Listener for JMX

The node agent uses JMX to communicate with the Domain Administration Server. Therefore it must have a port to listen on for JMX requests, and other listener information.

1 **In the tree component, expand the Node Agents node.**

2 **Select the node agent configuration to edit.**

3 **Click the JMX tab.**

4 **In the Address field, enter an IP address or host name.**

Enter `0.0.0.0` if the listener listens on all IP addresses for the server using a unique port value. Otherwise, enter a valid IP address for the server.

5 **In the Port field, type the port on which the node agent's JMX connector will listen.**

If the IP address is `0.0.0.0`, the port number must be unique.

6 **In the JMX Protocol field, type the protocol that the JMX connector supports.**

The default is rmi_jrmp.

7 **Click the checkbox next to Accept All Addresses to allow a connection to all IP addresses.**

The node agent listens on a specific IP address associated to a network card or listens on all IP addresses. Accepting all addresses puts the value `0.0.0.0` in the "listening host address" property.

8 **In the Realm Name field, type the name of the realm that handles authentication for the listener.**

In the Security section of this page, configure the listener to use SSL, TLS, or both SSL and TLS security.

To set up a secure listener, do the following:

9 **Check the Enabled box in the Security field.**

Security is enabled by default.

10 **Set client authentication.**

To require clients to authenticate themselves to the server when using this listener, check the Enabled box in the Client Authentication field.

11 **Enter a certificate nickname.**

Enter the name of an existing server keypair and certificate in the Certificate NickName field.

For information about working with certificates and SSL, see the Admin Console online help.

12 **In the SSL3/TLS section:**

a. **Check the security protocol(s) to enable on the listener.**

You must check either SSL3 or TLS, or both protocols.

b. **Check the cipher suite used by the protocol(s).**

To enable all cipher suites, check All Supported Cipher Suites.

13 **Click Save.**

◆ ◆ ◆ CHAPTER 8

Configuring High Availability Session Persistence and Failover

This chapter explains how to enable and configure high availability session persistence.

Overview of Session Persistence and Failover

Communications Application Server provides high availability session persistence through *failover* of HTTP/SIP session data and stateful session bean (SFSB) session data. Failover means that in the event of an server instance or hardware failure, another server instance takes over a distributed session.

Requirements

A distributed session can run in multiple Sun Java System Communications Application Server instances, if:

- Each server instance has access to the same session state data. Communications Application Server provides the following types of high availability storage for HTTP/SIP session and stateful session bean data:
  - In-memory replication on other servers in the cluster. In-memory replication is enabled by default with the cluster profile.

    The use of in-memory replication requires the Group Management Service (GMS) to be enabled. For more information about GMS, see "Group Management Service" on page 103.

If server instances in a cluster are located on different machines, ensure that the following prerequisites are met:

- To ensure that GMS and in-memory replication function correctly, the machines must be on the same subnet.
- To ensure that in-memory replication functions correctly, the system clocks on all machines in the cluster must be synchronized as closely as possible.

- Each server instance has the same distributable web application deployed to it. The `web-app` element of the `web.xml` deployment descriptor file or the `sip-app` element of the `sip.xml` deployment descriptor file must contain the `distributable` element.
- The web application uses high-availability session persistence. If a non-distributable web application is configured to use high-availability session persistence, the server writes an error to the log file.
- The web application must be deployed using the `deploy` or `deploydir` command with the `--availabilityenabled` option set to `true`. For more information on these commands, see `deploy`(1) and `deploydir`(1).

Restrictions

When a session fails over, any references to open files or network connections are lost. Applications must be coded with this restriction in mind.

You can only bind certain objects to distributed sessions that support failover. Contrary to the Servlet 2.4 specification, Sun Java System Communications Application Server does not throw an `IllegalArgumentException` if an object type not supported for failover is bound into a distributed session.

You can bind the following objects into a distributed session that supports failover:

- Local home and object references for all EJB components.
- Colocated stateless session, stateful session, or entity bean reference .
- Distributed stateless session, stateful session, or entity bean reference.
- JNDI Context for `InitialContext` and `java:comp/env`.
- `UserTransaction` objects. However, if the instance that fails is never restarted, any prepared global transactions are lost and might not be correctly rolled back or committed.
- Serializable Java types.

You cannot bind the following object types into sessions that support failover:

- JDBC DataSource
- Java Message Service (JMS) `ConnectionFactory` and `Destination` objects
- JavaMail™ Session
- Connection Factory

- Administered Objects
- Web service reference

In general, for these objects, failover will not work. However, failover might work in some cases, if for example the object is serializable.

Setting Up High Availability Session Persistence

This section explains how to set up high availability session persistence, with the following topics:

- "To Set Up High Availability Session Persistence" on page 143
- "Enabling Session Availability" on page 144

▼ To Set Up High Availability Session Persistence

Before You Begin High availability session persistence is incompatible with dynamic deployment, dynamic reloading, and autodeployment. These features are for development, not production environments, so you must disable them before enabling HA session persistence. For information about how to disable these features, see *Sun Java System Application Server 9.1 Application Deployment Guide*.

1. **Create an Communications Application Server cluster.**

   For more information, see "To Create a Cluster" on page 105 .

2. **Set up load balancing for the cluster.**

   For more information , see "Setting Up HTTP Load Balancing" on page 73 or Chapter 2, "Configuring Converged Load Balancing," for SIP or converged web/SIP applications.

3. **Enable availability for the desired application server instances and web or EJB containers.**

   Then configure the session persistence settings. Choose one of these approaches:

   - Use Admin Console. See "Enabling Availability for a Server Instance" on page 144.
   - Use the `asadmin` command-line utility. See `set(1)`.

4. **Restart each server instance in the cluster.**

   If the instance is currently serving requests, quiesce the instance before restarting it so that the instance gets enough time to serve the requests it is handling. For more information, see "Disabling (Quiescing) a Server Instance or Cluster" on page 84

5. **Enable availability for any specific SFSB that requires it.**

   Select methods for which checkpointing the session state is necessary. See "Configuring Availability for an Individual Bean" on page 154

6 Make each web/SIP module distributable if you want it to be highly available.

7 Enable availability for individual applications, web/SIP modules, or EJB modules during deployment.

See "Configuring Availability for an Individual Application or EJB Module" on page 154

In the Administration Console, check the Availability Enabled box, or use the `asadmin deploy` command with the `--availabilityenabled` option set to `true`.

Enabling Session Availability

You can enable session availability at five different scopes (from highest to lowest):

1. Server instance, enabled by default. Enabling session availability for the server instance means that all applications running on the server instance can have high-availability session persistence. For instructions, see next section, "Enabling Availability for a Server Instance" on page 144 .
2. Container (web/SIP or EJB), enabled by default. For information on enabling availability at the container level, see:
   - "Configuring Availability for the Web Container" on page 145
   - "Configuring Availability for the SIP Container" on page 148
   - "Configuring Availability for the EJB Container" on page 153
3. Application, disabled by default.
4. Standalone web/SIP or EJB module, disabled by default.
5. Individual SFSB, disabled by default.

To enable availability at a given scope, you must enable it at all higher levels as well. For example, to enable availability at the application level, you must also enable it at the server instance and container levels.

The default for a given level is the setting at the next level up. For example, if availability is enabled at the container level, it is enabled by default at the application level.

When availability is disabled at the server instance level, enabling it at any other level has no effect. When availability is enabled at the server instance level, it is enabled at all levels unless explicitly disabled.

Enabling Availability for a Server Instance

To enable availability for a server instance, use the `asadmin set` command to set the configuration's `availability-service.availability-enabled` property to true.

For example, if `config1` is the configuration name:

```
asadmin set --user admin --passwordfile password.txt
--host localhost
--port 4849
config1.availability-service.availability-enabled="true"
```

▼ To Enable Availability for the Server Instance with Admin Console

1. **In the tree component, expand the Configurations node.**

2. **Expand the node for the configuration you want to edit.**

3. **Select the Availability Service node.**

4. **In the Availability Service page, enable instance level availability by checking the Availability Service box.**

   To disable it, uncheck the box.

5. **Click on the Save button.**

6. **Stop and restart the server instance.**

HTTP/SIP Session Failover

Java EE applications typically have significant amounts of session state data. A web shopping cart is the classic example of session state. Also, an application can cache frequently-needed data in the session object. In fact, almost all applications with significant user interactions need to maintain session state.

Configuring Availability for the Web Container

Note – If you are using in-memory replication to store session state data, you *must* use the `asadmin set` command to enable web container availability and to set properties.

For example, use the `set` command as follows, where `config1` is the configuration name:

```
asadmin set
config1.availability-service.web-container-availability.availability-enabled="true"
```

```
asadmin set
config1.availability-service.web-container-availability.persistence-frequency="time-based"
```

▼ To Enable Availability for the Web Container with Admin Console

1. **In the tree component, select the desired configuration.**

2. **Click on Availability Service.**

3. **Select the Web Container Availability tab.**

   Check the Availability Service box to enable availability. To disable it, uncheck the box.

4. **Change other settings, as described in the following section, "Web Container Availability Settings" on page 146**

5. **Restart the server instance.**

Web Container Availability Settings

The Web Container Availability tab of the Availability Service enables you to change these availability settings:

Persistence Type: Specifies the session persistence mechanism for web applications that have availability enabled. Allowed values are `memory` (no persistence) `file` (the file system), and `replicated` (memory on other servers).

If web container availability is enabled, the default persistence type depends on the profile, as shown in the following table.

Profile	Persistence Type
Developer	`memory`
Cluster	`replicated`

For production environments that require session persistence, use `replicated`. The `memory` persistence type and the `file` persistence type do not provide high availability session persistence.

If web container availability is disabled, the default persistence type `memory`.

Persistence Frequency: Specifies how often the session state is stored. Applicable only if the Persistence Type is `replicated`. Allowed values are:

- `web-method` - The session state is stored at the end of each web request prior to sending a response back to the client. This mode provides the best guarantee that the session state is fully updated in case of failure. This is the default.

- `time-based` - The session state is stored in the background at the frequency set by the `reapIntervalSeconds` store property. This mode provides does not guarantee that session state is fully updated. However, it can provide a significant performance improvement because the state is not stored after each request.

Persistence Scope : Specifies how much of the session object and how often session state is stored. Applicable only if the Persistence Type is `replicated`. Allowed values are as follows:

- `session` - The entire session state is stored every time. This mode provides the best guarantee that your session data is correctly stored for any distributable web application. This is the default.
- `modified-session` - The entire session state is stored if it has been modified. A session is considered to have been modified if `HttpSession.setAttribute()` or `HttpSession.removeAttribute()` was called. You must guarantee that `setAttribute()` is called every time an attribute is changed. This is not a Java EE specification requirement, but it is required for this mode to work properly.
- `modified-attribute` - Only modified session attributes are stored. For this mode to work properly, you must follow a few guidelines:
  - Call `setAttribute()` every time the session state is modified.
  - Make sure there are no cross-references between attributes. The object graph under each distinct attribute key is serialized and stored separately. If there are any object cross references between the objects under each separate key, they are not serialized and deserialized correctly.
  - Distribute the session state across multiple attributes, or at least between a read-only attribute and a modifiable attribute.

Single Sign-On State: Check this box to enable persistence of the single sign-on state. To disable it, uncheck the box. For more information, see "Using Single Sign-on with Session Failover" on page 150

Configuring Availability for Individual Web Applications

To enable and configure availability for an individual web or converged web/SIP application, edit the application deployment descriptor file, `sun-web.xml`. The settings in an application's deployment descriptor override the web container's availability settings.

The `session-manager` element's `persistence-type` attribute determines the type of session persistence an application uses. It must be set to `replicated` to enable high availability session persistence.

For more information about the `sun-web.xml` file, see "The sun-web.xml File" in *Sun Java System Application Server 9.1 Application Deployment Guide*.

Example

```
<sun-web-app> ...
  <session-config>
    <session-manager persistence-type="replicated">
      <manager-properties>
        <property name="persistenceFrequency" value="web-method" />
      </manager-properties>
      <store-properties>
        <property name="persistenceScope" value="session" />
      </store-properties>
    </session-manager> ...
</session-config> ...
```

Configuring Availability for the SIP Container

Use the `asadmin set` command to enable web container availability and to change availability settings. For example, use the `set` command as follows, where `config1` is the configuration name:

```
asadmin set
config1.availability-service.sip-container-availability.availability-enabled="true"
```

```
asadmin set
config1.availability-service.sip-container-availability.persistence-frequency="time-based"
```

▼ To Enable Availability for the SIP Container with Admin Console

1. **In the tree component, select the desired configuration.**
2. **Click on Availability Service.**
3. **Select the SIP Container Availability tab.**

   Check the Availability Service box to enable availability. To disable it, uncheck the box.
4. **Change other settings, as described in the following section, "SIP Container Availability Settings" on page 148.**
5. **Restart the server instance.**

SIP Container Availability Settings

The SIP Container Availability tab of the Availability Service enables you to change these availability settings:

Persistence Type: Specifies the session persistence mechanism for web applications that have availability enabled. Allowed values are `memory` (no persistence) and `replicated` (memory on other servers). If SIP container availability is disabled, the default persistence type `memory`. If SIP container availability is enabled, the default persistence type is `replicated`. For production environments that require session persistence, use `replicated`. The `memory` persistence type does not provide high availability session persistence.

Persistence Frequency: Specifies how often the session state is stored. Applicable only if the Persistence Type is `replicated`. Allowed values are:

- `sip-transaction` - The session state is stored at the end of each request prior to sending a response back to the client. This mode provides the best guarantee that the session state is fully updated in case of failure. This is the default.
- `time-based` - The session state is stored in the background at the frequency set by the `reapIntervalSeconds` store property. This mode provides does not guarantee that session state is fully updated. However, it can provide a significant performance improvement because the state is not stored after each request.

Persistence Scope : Specifies how much of the session object and how often session state is stored. Applicable only if the Persistence Type is `replicated`. Allowed values are as follows:

- `session` - The entire session state is stored every time. This mode provides the best guarantee that your session data is correctly stored for any distributable web application. This is the default.
- `modified-session` - The entire session state is stored if it has been modified. A session is considered to have been modified if `SipApplicationSession.setAttribute()` or `SipApplicationSession.removeAttribute()` was called. You must guarantee that `setAttribute()` is called every time an attribute is changed. This is not a Java EE specification requirement, but it is required for this mode to work properly.
- `modified-attribute` - Only modified session attributes are stored. For this mode to work properly, you must follow a few guidelines:
  - Call `setAttribute()` every time the session state is modified.
  - Make sure there are no cross-references between attributes. The object graph under each distinct attribute key is serialized and stored separately. If there are any object cross references between the objects under each separate key, they are not serialized and deserialized correctly.
  - Distribute the session state across multiple attributes, or at least between a read-only attribute and a modifiable attribute.

Single Sign-On State: Check this box to enable persistence of the single sign-on state. To disable it, uncheck the box. For more information, see "Using Single Sign-on with Session Failover" on page 150. **[Remark 8–4 : Applicable to SIP?]**

Configuring Availability for Individual SIP Applications

To enable and configure availability for an individual SIP application, edit the application deployment descriptor file, `sun-sip.xml`. The settings in an application's deployment descriptor override the SIP container's availability settings.

For a converged web/SIP application, edit `sun-web.xml` to configure the web container and `sun-sip.xml` to configure the SIP container.

The `session-manager` element's `persistence-type` attribute determines the type of session persistence an application uses. It must be set to `replicated` to enable high availability session persistence.

For more information about the `sun-sip.xml` file, see "The sun-sip.xml File" in *Sun Java System Application Server 9.1 Application Deployment Guide*.

Example

```
<sun-web-app> ...
  <session-config>
    <session-manager persistence-type="replicated">
      <manager-properties>
        <property name="persistenceFrequency" value="sip-transaction" />
      </manager-properties>
      <store-properties>
        <property name="persistenceScope" value="session" />
      </store-properties>
    </session-manager> ...
</session-config> ...
```

Using Single Sign-on with Session Failover

In a single application server instance, once a user is authenticated by an application, the user is not required to re-authenticate individually to other applications running on the same instance. This is called *single sign-on*. For more information, see "User Authentication for Single Sign-on" in *Sun Java System Application Server 9.1 Developer's Guide*.

For this feature to continue to work even when an HTTP/SIP session fails over to another instance in a cluster, single sign-on information must be persisted using in-memory replication. To persist single sign-on information, first, enable availability for the server instance and the web container, then enable single-sign-on state failover. **[Remark 8–5 : Is replicated persistence sufficient?]**

You can enable single sign-on state failover with the Admin Console in the Web Container Availability or SIP Container Availability tab of the Availability Service, as described in "Configuring Availability for the Web Container" on page 145. You can also use the asadmin set command to set the configuration's availability-service.web-container-availability.sso-failover-enabled property to true.

For example, use the set command as follows, where config1 is the configuration name:

```
asadmin set --user admin --passwordfile password.txt
--host localhost --port 4849
config1.availability-service.web-container-availability.
sso-failover-enabled="true"
```

Single Sign-On Groups

Applications that can be accessed through a single name and password combination constitute a *single sign-on group*. For HTTP/SIP sessions corresponding to applications that are part of a single sign-on group, if one of the sessions times out, other sessions are not invalidated and continue to be available. This is because time out of one session should not affect the availability of other sessions.

As a corollary of this behavior, if a session times out and you try to access the corresponding application from the same browser window that was running the session, you are not required to authenticate again. However, a new session is created.

Take the example of a shopping cart application that is a part of a single sign-on group with two other applications. Assume that the session time out value for the other two applications is higher than the session time out value for the shopping cart application. If your session for the shopping cart application times out and you try to run the shopping cart application from the same browser window that was running the session, you are not required to authenticate again. However, the previous shopping cart is lost, and you have to create a new shopping cart. The other two applications continue to run as usual even though the session running the shopping cart application has timed out.

Similarly, suppose a session corresponding to any of the other two applications times out. You are not required to authenticate again while connecting to the application from the same browser window in which you were running the session.

Note – This behavior applies only to cases where the session times out. If single sign-on is enabled and you invalidate one of the sessions using `HttpSession.invalidate()` or `SipApplicationSession.invalidate()`, the sessions for all applications belonging to the single sign-on group are invalidated. If you try to access any application belonging to the single sign-on group, you are required to authenticate again, and a new session is created for the client accessing the application.

Stateful Session Bean Failover

Stateful session beans (SFSBs) contain client-specific state. There is a one-to-one relationship between clients and the stateful session beans. At creation, the EJB container gives each SFSB a unique session ID that binds it to a client.

An SFSB's state can be saved in a persistent store in case a server instance fails. The state of an SFSB is saved to the persistent store at predefined points in its life cycle. This is called *checkpointing*. If enabled, checkpointing generally occurs after the bean completes any transaction, even if the transaction rolls back.

However, if an SFSB participates in a bean-managed transaction, the transaction might be committed in the middle of the execution of a bean method. Since the bean's state might be undergoing transition as a result of the method invocation, this is not an appropriate time to checkpoint the bean's state. In this case, the EJB container checkpoints the bean's state at the end of the corresponding method, provided the bean is not in the scope of another transaction when that method ends. If a bean-managed transaction spans across multiple methods, checkpointing is delayed until there is no active transaction at the end of a subsequent method.

The state of an SFSB is not necessarily transactional and might be significantly modified as a result of non-transactional business methods. If this is the case for an SFSB, you can specify a list of checkpointed methods, as described in "Specifying Methods to Be Checkpointed" on page 155

If a distributable web application references an SFSB, and the web application's session fails over, the EJB reference is also failed over.

If an SFSB that uses session persistence is undeployed while the Communications Application Server instance is stopped, the session data in the persistence store might not be cleared. To prevent this, undeploy the SFSB while the Communications Application Server instance is running.

Configuring Availability for the EJB Container

▼ To Enable Availability for the EJB Container

1. **Select the EJB Container Availability tab.**

2. **Check the Availability Service box.**

   To disable availability, uncheck the box.

3. **Change other settings as described in "Availability Settings" on page 154**

4. **Click on the Save button.**

5. **Restart the server instance.**

More Information

Equivalent asadmin command

To enable availability for the EJB container use the `asadmin set` command to set the following three properties for the configuration:

- `availability-service.ejb-container-availability.availability-enabled`
- `availability-service.ejb-container-availability.sfsb-persistence-type`
- `availability-service.ejb-container-availability.sfsb-ha-persistence-type`

For example, if `config1` is the configuration name, use the following commands:

```
asadmin set --user admin --passwordfile password.txt
--host localhost
--port 4849
config1.availability-service.
ejb-container-availability.availability-enabled="true"

asadmin set --user admin --passwordfile password.txt --host localhost --port
4849
config1.availability-service.
ejb-container-availability.sfsb-persistence-type="file"

asadmin set --user admin --passwordfile password.txt
--host localhost
--port 4849
config1.availability-service.
ejb-container-availability.sfsb-ha-persistence-type="replicated"
```

Availability Settings

The EJB Container Availability tab of the Availability Service enables you to change these settings:

HA Persistence Type: Specifies the session persistence and passivation mechanism for SFSBs that have availability enabled. Allowed values are `file` (the file system), and `replicated` (memory on other servers). The default value is `replicated`. For production environments that require session persistence, use `replicated`.

SFSB Persistence Type: Specifies the passivation mechanism for SFSBs that *do not* have availability enabled. Allowed values are `file` (the default) and `replicated`.

If either Persistence Type is set to `file`, the EJB container specifies the file system location where the passivated session bean state is stored. Checkpointing to the file system is useful for testing but is not for production environments. For information about configuring store properties, see the Admin Console online help.

HA persistence enables a cluster of server instances to recover the SFSB state if any server instance fails. The HA store is also used as the passivation and activation store. Use this option in a production environment that requires SFSB state persistence.

Configuring the SFSB Session Store When Availability Is Disabled

If availability is disabled, the local file system is used for SFSB state passivation, but not persistence. To change where the SFSB state is stored, change the Session Store Location setting in the EJB container. For information about configuring store properties, see the Admin Console online help.

Configuring Availability for an Individual Application or EJB Module

You can enable SFSB availability for an individual application or EJB module during deployment:

- If you are deploying with the Admin Console, check the Availability Enabled checkbox.
- If you are deploying using use the `asadmin deploy` or `asadmin deploydir` commands, set the `--availabilityenabled` option to `true`. For more information, see `deploy`(1) and `deploydir`(1).

Configuring Availability for an Individual Bean

To enable availability and select methods to be checkpointed for an individual SFSB, use the `sun-ejb-jar.xml` deployment descriptor file. .

To enable high availability session persistence, set `availability-enabled="true"` in the `ejb` element. To control the size and behavior of the SFSB cache, use the following elements:

- `max-cache-size` : specifies the maximum number of session beans that are held in cache. If the cache overflows (the number of beans exceeds `max-cache-size`), the container then passivates some beans or writes out the serialized state of the bean into a file. The directory in which the file is created is obtained from the EJB container using the configuration APIs.
- `resize-quantity`
- `cache-idle-timeout-in-seconds`
- `removal-timeout-in-seconds`
- `victim-selection-policy`

For more information about `sun-ejb-jar.xml`, see “The sun-ejb-jar.xml File” in *Sun Java System Application Server 9.1 Application Deployment Guide*.

EXAMPLE 8–1 Example of an EJB Deployment Descriptor With Availability Enabled

```
<sun-ejb-jar>
    ...
    <enterprise-beans>
        ...
        <ejb availability-enabled="true">
            <ejb-name>MySFSB</ejb-name>
        </ejb>
        ...
    </enterprise-beans>
</sun-ejb-jar>
```

Specifying Methods to Be Checkpointed

If enabled, checkpointing generally occurs after the bean completes any transaction, even if the transaction rolls back. To specify additional optional checkpointing of SFSBs at the end of non-transactional business methods that cause important modifications to the bean’s state, use the `checkpoint-at-end-of-method` element in the `ejb` element of the `sun-ejb-jar.xml` deployment descriptor file.

The non-transactional methods in the `checkpoint-at-end-of-method` element can be:

- `create()` methods defined in the home interface of the SFSB, if you want to checkpoint the initial state of the SFSB immediately after creation
- For SFSBs using container managed transactions only, methods in the remote interface of the bean marked with the transaction attribute TX_NOT_SUPPORTED or TX_NEVER
- For SFSBs using bean managed transactions only, methods in which a bean managed transaction is neither started nor committed

Any other methods mentioned in this list are ignored. At the end of invocation of each of these methods, the EJB container saves the state of the SFSB to persistent store.

Note – If an SFSB does not participate in any transaction, and if none of its methods are explicitly specified in the `checkpoint-at-end-of-method` element, the bean's state is not checkpointed at all even if `availability-enabled="true"` for this bean.

For better performance, specify a *small* subset of methods. The methods should accomplish a significant amount of work or result in important modification to the bean's state.

EXAMPLE 8–2 Example of EJB Deployment Descriptor Specifying Methods Checkpointing

```
<sun-ejb-jar>
    ...
    <enterprise-beans>
        ...
        <ejb availability-enabled="true">
            <ejb-name>ShoppingCartEJB</ejb-name>
            <checkpoint-at-end-of-method>
                <method>
                    <method-name>addToCart</method-name>
                </method>
            </checkpoint-at-end-of-method>
        </ejb>
        ...
    </enterprise-beans>
</sun-ejb-jar>
```

CHAPTER 9

Java Message Service Load Balancing and Failover

This chapter describes how to configure load balancing and failover of the Java Message Service (JMS) for use with the Communications Application Server. It contains the following topics:

Overview of Java Message Service

The Java Message Service (JMS) API is a messaging standard that allows Java EE applications and components to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous. The Sun Java System Message Queue (MQ), which implements JMS, is tightly integrated with Communications Application Server, enabling you to create components such as message-driven beans (MDBs).

MQ is integrated with Communications Application Server using a *connector module,* also known as a resource adapter, defined by the Java EE Connector Architecture Specification 1.5. Java EE components deployed to the Communications Application Server exchange JMS messages using the JMS provider integrated via the connector module. Creating a JMS resource in Communications Application Server creates a connector resource in the background. So, each JMS operation invokes the connector runtime and uses the MQ resource adapter in the background.

You can manage the Java Message Service through the Admin Console or the `asadmin` command-line utility.

Further Information

For more information on configuring JMS resources, see Chapter 4, “Configuring Java Message Service Resources,” in *Sun Java System Application Server 9.1 Administration Guide*. For more information on JMS, see Chapter 18, “Using the Java Message Service,” in *Sun Java System Application Server 9.1 Developer’s Guide*. For more information on connectors (resource adapters), see Chapter 12, “Developing Connectors,” in *Sun Java System Application Server 9.1 Developer’s Guide*.

For more information about the Sun Java System Message Queue, see the Message Queue documentation (`http://docs.sun.com/coll/1343.4`). For general information about the JMS API, see the JMS web page (`http://java.sun.com/products/jms/index.html`)

Configuring the Java Message Service

The Java Message Service configuration is available to all inbound and outbound connections to the Sun Java System Communications Application Server cluster or instance. You can configure the Java Message Service with:

- The Administration Console. Open the Java Message Service component under the relevant configuration. For details, see Chapter 4, “Configuring Java Message Service Resources,” in *Sun Java System Application Server 9.1 Administration Guide*.
- The `asadmin set` command. You can set the following attributes:

  ```
  server.jms-service.init-timeout-in-seconds = 60
  server.jms-service.type = LOCAL
  server.jms-service.start-args =
  server.jms-service.default-jms-host = default_JMS_host
  server.jms-service.reconnect-interval-in-seconds = 60
  server.jms-service.reconnect-attempts = 3
  server.jms-service.reconnect-enabled = true
  server.jms-service.addresslist-behavior = random
  server.jms-service.addresslist-iterations = 3
  server.jms-service.mq-scheme = mq
  server.jms-service.mq-service = jms
  ```

  You can also set these properties:

  ```
  server.jms-service.property.instance-name = imqbroker
  server.jms-service.property.instance-name-suffix =
  server.jms-service.property.append-version = false
  ```

  Use the `asadmin getcommand` to list all the Java Message Service attributes and properties. For more information on `asadmin get`, see `get`(1). For more information on `asadmin set`, see `set`(1).

You can override the Java Message Service configuration using JMS connection factory settings. For details, see "JMS Connection Factories" in *Sun Java System Application Server 9.1 Administration Guide*.

Note – You must restart the Communications Application Server instance after changing the configuration of the Java Message Service.

For more information about JMS administration, see Chapter 4, "Configuring Java Message Service Resources," in *Sun Java System Application Server 9.1 Administration Guide*

Java Message Service Integration

MQ can be integrated with Communications Application Server in three ways: LOCAL, REMOTE, and EMBEDDED. These modes are represented in the Admin Console by the Java Message Service Type attribute.

LOCAL Java Message Service

When the Type attribute is LOCAL (the default for cluster instances), the Communications Application Server will start and stop the MQ broker specified as the Default JMS host. The MQ process is started out-of-process, in a separate VM, from the Application Server process. Application Server supplies an additional port to the broker . This port will be used by the broker to start the RMI registry. This port number will be equal to the configured JMS port for that instance plus 100. For example, if the JMS port number is 37676, then this additional port number will be 37776.

To create a one-to-one relationship between Communications Application Server instances and Message Queue brokers, set the type to LOCAL and give each Communications Application Server instance a different default JMS host. You can do this regardless of whether clusters are defined in the Communications Application Server or MQ.

With LOCAL type, use the Start Arguments attribute to specify MQ broker startup parameters.

REMOTE Java Message Service

When the Type attribute is REMOTE, the MQ broker must be started separately. For information about starting the broker, see the *Sun Java System Message Queue Administration Guide*.

In this case, Communications Application Server will use an externally configured broker or broker cluster. Also, you must start and stop MQ brokers separately from Communications Application Server, and use MQ tools to configure and tune the broker or broker cluster. REMOTE type is most suitable for Communications Application Server clusters.

With REMOTE type, you must specify MQ broker startup parameters using MQ tools. The Start Arguments attribute is ignored.

EMBEDDED Java Message Service

When the JMS Type attribute is EMBEDDED, it means that the application server and the JMS broker are co-located in the same VM and the JMS service is started in-process and managed by the Application Server. In this mode, the JMS operations by pass the networking stack leading to performance optimization.

JMS Hosts List

A JMS host represents an MQ broker. The Java Message Service contains a *JMS Hosts list* (also called `AddressList`) that contains all the JMS hosts that Communications Application Server uses.

The JMS Hosts list is populated with the hosts and ports of the specified MQ brokers and is updated whenever a JMS host configuration changes. When you create JMS resources or deploy MDBs, they inherit the JMS Hosts list.

Note – In the Sun Java System Message Queue software, the `AddressList` property is called `imqAddressList`.

Default JMS Host

One of the hosts in the JMS Hosts list is designated the default JMS host, named `Default_JMS_host`. The Communications Application Server instance starts the default JMS host when the Java Message Service type is configured as LOCAL.

If you have created a multi-broker cluster in the Sun Java System Message Queue software, delete the default JMS host, then add the Message Queue cluster's brokers as JMS hosts. In this case, the default JMS host becomes the first one in the JMS Hosts list.

When the Communications Application Server uses a Message Queue cluster, it executes Message Queue specific commands on the default JMS host. For example, when a physical destination is created for a Message Queue cluster of three brokers, the command to create the physical destination is executed on the default JMS host, but the physical destination is used by all three brokers in the cluster.

Creating JMS Hosts

You can create additional JMS hosts in the following ways:

- Use the Administration Console. Open the Java Message Service component under the relevant configuration, select the JMS Hosts component, and then click New. For more information, see the Admin Console online help.

- Use the `asadmin` `create-jms-host` command. For details, see `create-jms-host`(1).

  The JMS Hosts list is updated whenever a JMS host configuration changes.

Connection Pooling and Failover

Communications Application Server supports JMS connection pooling and failover. The Sun Java System Communications Application Server pools JMS connections automatically. When the Address List Behavior attribute is `random` (the default), Communications Application Server selects its primary broker randomly from the JMS host list. When failover occurs, MQ transparently transfers the load to another broker and maintains JMS semantics. The default value for the Address List Behavior attribute is `priority`, if the JMS type is of type LOCAL.

To specify whether the Communications Application Server tries to reconnect to the primary broker when the connection is lost, select the Reconnect checkbox. If enabled and the primary broker goes down, Communications Application Server tries to reconnect to another broker in the JMS Hosts list.

When Reconnect is enabled, also specify the following attributes:

- **Address List Behavior**: whether connection attempts are in the order of addresses in the JMS Hosts List (priority) or random order (random). If set to Priority, Java Message Service tries to connect to the first MQ broker specified in the JMS Hosts list and uses another one only if the first broker is not available. If set to Random, Java Message Service selects the MQ broker randomly from the JMS Hosts list. If there are many clients attempting a connection using the same connection factory, use this setting to prevent them from all attempting to connect to the same address.
- **Address List Iterations**: number of times the Java Message Service iterates through the JMS Hosts List in an effort to establish (or re-establish) a connection). A value of -1 indicates that the number of attempts is unlimited.
- **Reconnect Attempts**: the number of attempts to connect (or reconnect) for each address in the JMS hosts list before the client runtime tries the next address in the list. A value of -1 indicates that the number of reconnect attempts is unlimited (the client runtime attempts to connect to the first address until it succeeds).
- **Reconnect Interval**: number of seconds between reconnect attempts. This applies for attempts on each address in the JMS hosts list 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.

You can override these settings using JMS connection factory settings. For details, see “JMS Connection Factories” in *Sun Java System Application Server 9.1 Administration Guide*.

Load-Balanced Message Inflow

You can configure `ActivationSpec` properties of the `jmsra` resource adapter in the `sun-ejb-jar.xml` file for a message-driven bean using `activation-config-property` elements. Whenever a message-driven bean (`EndPointFactory`) is deployed, the connector runtime engine finds these properties and configures them accordingly in the resource adapter. See “activation-config-property” in *Sun Java System Application Server 9.1 Application Deployment Guide*.

The Communications Application Server transparently enables messages to be delivered randomly to message-driven beans having the same `ClientID` . The `ClientID` is required for durable subscribers.

For non-durable subscribers in which the `ClientID` is not configured, all instances of a specific message-driven bean that subscribe to same topic are considered equal. When a message-driven bean is deployed to multiple instances of the Communications Application Server, only one of the message-driven beans receives the message. If multiple distinct message-driven beans subscribe to same topic, one instance of each message-driven bean receives a copy of the message.

To support multiple consumers using the same queue, set the `maxNumActiveConsumers` property of the physical destination to a large value. If this property is set, the Sun Java System Message Queue software allows up to that number of message-driven beans to consume messages from same queue. The message is delivered randomly to the message-driven beans. If `maxNumActiveConsumers` is set to `-1`, there is no limit to the number of consumers.

To ensure that local delivery is preferred, set `addresslist-behavior` to priority. This setting specifies that the first broker in the `AddressList` is selected first. This first broker is the local colocated Message Queue instance. If this broker is unavailable, connection attempts are made to brokers in the order in which they are listed in the `AddressList`. This setting is the default for Communications Application Server instances that belong to a cluster.

Note – Clustering features are not available in the developer profile. For information about profiles, see “Usage Profiles” in *Sun Java System Application Server 9.1 Administration Guide*.

Using MQ Clusters with Application Server

MQ Enterprise Edition supports multiple interconnected broker instances known as a *broker cluster*. With broker clusters, client connections are distributed across all the brokers in the cluster. Clustering provides horizontal scalability and improves availability.

This section describes how to configure Communications Application Server to use Sun Java System Message Queue clusters. It explains how to start and configure Message Queue clusters.

For more information about the topology of Communications Application Server and MQ deployment, see “Planning Message Queue Broker Deployment” in *Sun Java System Application Server 9.1 Deployment Planning Guide*.

Auto-clustering for non-HA Clusters

Till now, the administrator had to set up 'non-Highly Available' MQ clusters (MQ clusters with a master broker) separately as explained in the procedure following this section. In this release, in addition to the manual process (of type REMOTE) of setting up an MQ cluster, Application Server provides 'auto-clustering,' which means that a co-located non-HA cluster (of type LOCAL) will be created automatically when a user creates an Application Server cluster. This will be the default mode of creating MQ clusters. For example, when the administrator creates an Application Server cluster with three Application Server instances, each Application Server instance will be configured to work with a co-located broker, and as a result the three MQ broker instances will be made to form an MQ cluster transparently. The first Application Server instance's MQ broker will be set to be the master broker. Auto-clustering, however, has a disadvantage too. If the administrator adds an instance to the cluster, the MQ broker instance created automatically will not be able to take part in the cluster. This behavior also applies if an instance is removed from the cluster.

▼ To Enable MQ Clusters with Application Server Clusters

Before You Begin Perform the following steps if the cluster is of type REMOTE. If the cluster is of type LOCAL, steps 1 to 4 are not applicable .

1 **Create an Application Server cluster, if one does not already exist.**

For information on creating clusters, see “To Create a Cluster” on page 105.

2 **Create an MQ broker cluster.**

First, delete the default JMS host that refers to the broker started by the Domain Administration Server, and then create three external brokers (JMS hosts) that will be in the MQ broker cluster.

Create a JMS host with either the Admin Console or the `asadmin` command-line utility.

To use `asadmin`, the commands are for example:

```
asadmin delete-jms-host --target cluster1 default_JMS_host
asadmin create-jms-host --target cluster1
     --mqhost myhost1 --mqport 6769
     --mquser admin --mqpassword admin broker1
asadmin create-jms-host --target cluster1
     --mqhost myhost2 --mqport 6770
     --mquser admin --mqpassword admin broker2
asadmin create-jms-host --target cluster1
     --mqhost myhost3 --mqport 6771
     --mquser admin --mqpassword admin broker3
```

To create the hosts with Admin Console:

a. **Navigate to the JMS Hosts node (Configurations >** *config-name* **> Java Message Service > JMS Hosts)**

b. **Delete the default broker (default_JMS_host).**

   Select the checkbox next to it, and then click Delete.

c. **Click New to create each JMS host and enter its property values.**

   Fill in the values for host name, DNS name or IP address, port number, administrative user name and password.

3 **Start the master MQ broker and the other MQ brokers.**

In addition to the three external brokers started on JMS host machines, start one master broker on any machine. This master broker need not be part of a broker cluster. For example:

```
/usr/bin/imqbrokerd -tty -name brokerm -port 6772
 -cluster myhost1:6769,myhost2:6770,myhost2:6772,myhost3:6771
 -D"imq.cluster.masterbroker=myhost2:6772"
```

4 **Start the Application Server instances in the cluster.**

5 **Create JMS resources on the cluster:**

a. **Create JMS physical destinations.**

   For example, using `asadmin`:

```
asadmin create-jmsdest --desttype queue --target cluster1 MyQueue
asadmin create-jmsdest --desttype queue --target cluster1 MyQueue1
```

   To use Admin Console:

   i. **Navigate to the JMS Hosts page (Configurations >** *config-name* **> Java Message Service > Physical Destinations).**

   ii. **Click New to create each JMS physical destination.**

   iii. **For each destination, enter its name and type (queue).**

b. **Create JMS connection factories.**

   For example, using `asadmin`:

   ```
   asadmin create-jms-resource --target cluster1
   --restype javax.jms.QueueConnectionFactory jms/MyQcf
   asadmin create-jms-resource --target cluster1
   --restype javax.jms.QueueConnectionFactory jms/MyQcf1
   ```

   To use Admin Console:

   i. **Navigate to the JMS Connection Factories page (Resources > JMS Resources > Connection Factories).**

   ii. **To create each connection factory, click New.**

      The Create JMS Connection Factory page opens.

   iii. **For each connection factory, enter JNDI Name (for example `jms/MyQcf`) and Type, `javax.jms.QueueConnectionFactory`**

   iv. **Select the cluster from the list of available targets at the bottom of the page and click Add.**

   v. **Click OK to create the connection factory.**

c. **Create JMS destination resources.**

   For example, using `asadmin`:

   ```
   asadmin create-jms-resource --target cluster1
        --restype javax.jms.Queue
        --property imqDestinationName=MyQueue jms/MyQueue
   asadmin create-jms-resource --target cluster1
        --restype javax.jms.Queue
        --property imqDestinationName=MyQueue1 jms/MyQueue1
   ```

   To use Admin Console:

   i. **Navigate to the JMS Destination Resources page (Resources > JMS Resources > Connection Factories).**

   ii. **To create each destination resource, click New.**

      The Create JMS Destination Resource page opens.

   iii. **For each destination resource, enter JNDI Name (for example `jms/MyQueue`) and Type `javax.jms.Queue`.**

   iv. **Select the cluster from the list of available targets at the bottom of the page and click Add.**

v. **Click OK to create the destination resource.**

6 **Deploy the applications with the –** `retrieve` **option for application clients. For example:**

```
asadmin deploy --target cluster1
--retrieve /opt/work/MQapp/mdb-simple3.ear
```

7 **Access the application and test it to ensure it is behaving as expected.**

8 **If you want to return the Communications Application Server to its default JMS configuration, delete the JMS hosts you created and recreate the default. For example:**

```
asadmin delete-jms-host --target cluster1 broker1
asadmin delete-jms-host --target cluster1 broker2
asadmin delete-jms-host --target cluster1 broker3
asadmin create-jms-host --target cluster1
 --mqhost myhost1 --mqport 7676
 --mquser admin --mqpassword admin
 default_JMS_host
```

You can also perform the equivalent operation with Admin Console.

Troubleshooting If you encounter problems, consider the following:

- View the Application Server log file located at *as-install-dir*/nodeagents/*node-agent-name*/*instance-name*/logs/server.log. If you see in the log file that an MQ broker does not respond to a message, stop the broker and then restart it.
- View the broker log available at *as-install-dir*/nodeagents/*node-agent-name*/*instance-name*/imq/*imq-instance-name*/log/log.txt.
- For the Remote JMS type, always be sure to start MQ brokers first, and then the Application Server instances.
- When all MQ brokers are down, it takes 30 minutes for Application Server to go down or up, with the default values in Java Message Service. Tune Java Message Service values to get acceptable values for this timeout. For example:

  ```
  asadmin set --user admin --password administrator
  cluster1.jms-service.reconnect-interval-in-seconds=5
  ```

CHAPTER 10

RMI-IIOP Load Balancing and Failover

This chapter describes using Sun Java System Application Server's high-availability features for remote EJB references and JNDI objects over RMI-IIOP.

- "Overview" on page 167
- "Setting up RMI-IIOP Load Balancing and Failover" on page 169

Overview

With RMI-IIOP load balancing, IIOP client requests are distributed to different server instances or name servers. The goal is to spread the load evenly across the cluster, thus providing scalability. IIOP load balancing combined with EJB clustering and availability also provides EJB failover.

When a client performs a JNDI lookup for an object, the Naming Service creates a `InitialContext` (IC) object associated with a particular server instance. From then on, all lookup requests made using that IC object are sent to the same server instance. All `EJBHome` objects looked up with that `InitialContext` are hosted on the same target server. Any bean references obtained henceforth are also created on the same target host. This effectively provides load balancing, since all clients randomize the list of live target servers when creating `InitialContext` objects. If the target server instance goes down, the lookup or EJB method invocation will failover to another server instance.

IIOP load balancing and failover happens transparently. No special steps are needed during application deployment. IIOP load balancing and failover for the Communications Application Server supports dynamically reconfigured clusters. If the Communications Application Server instance on which the application client is deployed participates in a cluster, the Application Server finds all currently active IIOP endpoints in the cluster automatically. Therefore, you are not required to manually update the list of endpoints if a new instance is added to the cluster or deleted from the cluster. However, a client should have at least two endpoints specified for bootstrapping purposes, in case one of the endpoints has failed.

Requirements

Sun Java System Communications Application Server provides high availability of remote EJB references and `NameService` objects over RMI-IIOP, provided all the following apply:

- Your deployment has a cluster of at least two application server instances.
- Java EE applications are deployed to all application server instances and clusters that participate in load balancing.
- RMI-IIOP client applications are enabled for load balancing.

Communications Application Server supports load balancing for Java applications executing in the Application Client Container (ACC). See “Setting up RMI-IIOP Load Balancing and Failover” on page 169.

Note – Communications Application Server does not support RMI-IIOP load balancing and failover over secure sockets layer (SSL).

Algorithm

Communications Application Server uses a randomization and round-robin algorithm for RMI-IIOP load balancing and failover.

When an RMI-IIOP client first creates a new `InitialContext` object, the list of available Communications Application Server IIOP endpoints is randomized for that client. For that `InitialContext` object, the load balancer directs lookup requests and other `InitialContext` operations to the first endpoint on the randomized list. If the first endpoint is not available then the second endpoint in the list is used, and so on.

Each time the client subsequently creates a new `InitialContext` object, the endpoint list is rotated so that a different IIOP endpoint is used for `InitialContext` operations.

When you obtain or create beans from references obtained by an `InitialContext` object, those beans are created on the Communications Application Server instance serving the IIOP endpoint assigned to the `InitialContext` object. The references to those beans contain the IIOP endpoint addresses of all Communications Application Server instances in the cluster.

The *primary endpoint* is the bean endpoint corresponding to the `InitialContext` endpoint used to look up or create the bean. The other IIOP endpoints in the cluster are designated as *alternate endpoints*. If the bean's primary endpoint becomes unavailable, further requests on that bean fail over to one of the alternate endpoints.

You can configure RMI-IIOP load balancing and failover to work with applications running in the ACC.

Setting up RMI-IIOP Load Balancing and Failover

You can set up RMI-IIOP load balancing and failover for applications running in the application client container (ACC). Weighted round-robin load balancing is also supported.

▼ To Set Up RMI-IIOP Load Balancing for the Application Client Container

This procedure gives an overview of the steps necessary to use RMI-IIOP load balancing and failover with the application client container (ACC). For additional information on the ACC, see “Developing Clients Using the ACC” in *Sun Java System Application Server 9.1 Developer’s Guide*.

1 **Go to the *install_dir* `/bin` directory.**

2 **Run `package-appclient`.**

This utility produces an `appclient.jar` file. For more information on `package-appclient`, see `package-appclient`(1M).

3 **Copy the `appclient.jar` file to the machine where you want your client and extract it.**

4 **Edit the `asenv.conf` or `asenv.bat` path variables to refer to the correct directory values on that machine.**

The file is at *appclient-install-dir* `/config/`.

For a list of the path variables to update, see `package-appclient`(1M).

5 **If required, make the `appclient` script executable.**

For example, on UNIX use `chmod 700`.

6 **Find the IIOP listener port number of at least two instances in the cluster.**

You specify the IIOP listeners as endpoints in Step 7.

For each instance, obtain the IIOP listener port as follows:

a. **In the Admin Console’s tree component, expand the Clusters node.**

b. **Expand the cluster.**

c. **Select an instance in the cluster.**

d. **In the right pane, click the Properties tab.**

   e. **Note the IIOP listener port for the instance.**

7. **Add at least two `target-server` elements in the `sun-acc.xml` file.**

Note – Clustering features are not available in the developer profile. For information about profiles, see "Usage Profiles" in *Sun Java System Application Server 9.1 Administration Guide*.

Use the endpoints that you obtained in Step 6.

If the Communications Application Server instance on which the application client is deployed participates in a cluster, the ACC finds all currently active IIOP endpoints in the cluster automatically. However, a client should have at least two endpoints specified for bootstrapping purposes, in case one of the endpoints has failed.

The `target-server` element specifies one or more IIOP endpoints used for load balancing. The `address` attribute is an IPv4 address or host name, and the `port` attribute specifies the port number. See "client-container" in *Sun Java System Application Server 9.1 Application Deployment Guide*.

As an alternative to using `target-server` elements, you can use the `endpoints` property as follows:

```
jvmarg value = "-Dcom.sun.appserv.iiop.endpoints=host1:port1,host2:port2,..."
```

8. **If you require weighted round-robin load balancing, perform the following steps:**

   a. **Set the load-balancing weight of each server instance.**

   `asadmin set` *instance-name***`.lb-weight=`***weight*

   b. **In the `sun-acc.xml`, set the `com.sun.appserv.iiop.loadbalancingpolicy` property of the ACC to `ic-based-weighted`.**

```
...
<client-container send-password="true">
  <property name="com.sun.appserv.iiop.loadbalancingpolicy" value="ic-based-weighed"/>
...
```

9. **Deploy your client application with the `--retrieve` option to get the client jar file.**

   Keep the client jar file on the client machine.

   For example:

   `asadmin deploy --user admin --passwordfile pw.txt --retrieve /my_dir myapp`

10. **Run the application client as follows:**

    `appclient -client` *clientjar* `-name` *appname*

Example 10–1 Setting Load-Balancing Weights for RMI-IIOP Weighted Round-Robin Load Balancing

In this example, the load-balancing weights in a cluster of three instances are to be set as shown in the following table.

Instance Name	Load-Balancing Weight
`i1`	100
`i2`	200
`i3`	300

The sequence of commands to set these load balancing weights is as follows:

```
asadmin set i1.lb-weight=100
asadmin set i2.lb-weight=200
asadmin set i3.lb-weight=300
```

Next Steps To test failover, stop one instance in the cluster and see that the application functions normally. You can also have breakpoints (or sleeps) in your client application.

To test load balancing, use multiple clients and see how the load gets distributed among all endpoints.

Index