Sun GlassFish Communications Server 2.0 High Availability Administration Guide

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

Part No: 821–0196
September 2009

091113@22749

Contents

Tables

Examples

Preface

This book describes the high-availability features in Communications Server, including converged load balancing, HTTP load balancing, clusters, session persistence and failover

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

Sun GlassFish Communications Server Documentation Set

TABLE P–1 Books in the Communications Server Documentation Set

Book Title	Description
Documentation Center	Communications 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 Server product.
Installation Guide	Installing the software and its components.
Application Deployment Guide	Deployment of applications and application components to the Communications Server. Includes information about deployment descriptors.
Developer's Guide	Creating and implementing Java Platform, Enterprise Edition (Java EE platform) applications intended to run on the Communications Server that follow the open Java standards model for Java EE components and APIs. Includes information about developer tools, security, debugging, and creating lifecycle modules.
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 Server, including configuration, monitoring, security, resource management, and web services management.

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

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

Default Paths and File Names

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

TABLE P–2 Default Paths and File Names

Placeholder	Description	Default Value
as-install	Represents the base installation directory for Communications Server.	Solaris^{TM} and Linux installations, non-root user: *user's-home-directory*`/SUNWappserver` Solaris and Linux installations, root user: `/opt/SUNWappserver` Windows, all installations: *SystemDrive*`:\Sun\AppServer`
domain-root-dir	Represents the directory containing all domains.	All 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*
samples-dir	Represents the directory containing sample applications.	*as-install*`/samples`
docs-dir	Represents the directory containing documentation.	*as-install*`/docs`

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.

TABLE P–4 Symbol Conventions *(Continued)*

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

Documentation, Support, and Training

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

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

Third-Party Web Site References

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

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

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to `http://docs.sun.com` and click Feedback. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document.

CHAPTER 1

High Availability in Communications Server

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

This chapter contains the following topics.

- "Overview of High Availability" on page 17
- "How Communications ServerProvides High Availability" on page 20
- "Recovering from Failures" on page 22

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 Server provides the following high availability features:

- "Converged Load Balancer" on page 17
- "High Availability Session Persistence" on page 18
- "High Availability Java Message Service" on page 19
- "RMI-IIOP Load Balancing and Failover" on page 19

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. Communications Server provides the converged load balancer. A cluster (of Communications Server instances) or a standalone instance can be a dedicated load balancer. 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. The converged load balancer also helps achieve scalability of a system deployment. 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.

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

High Availability Session Persistence

Communications 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 Server provides the following types of storage for session state data:

- In-memory replication on other servers in the cluster

If the Communications 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 6, "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 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 7, "Java Message Service Load Balancing and Failover."

Connection Pooling and Failover

Communications Server supports JMS connection pooling and failover. The Communications Server pools JMS connections automatically. By default, Communications 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 123.

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 Communications Server" on page 124.

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 Communications Server instance on which the application client is deployed participates in a cluster, the Communications 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 8, "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 the Sun GlassFish Enterprise Server 2.1 Deployment Planning Guide. This manual also provides a high-level introduction to concepts such as:

- Communications 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 GlassFish Communications Server 2.0 Developer's Guide*.

Tuning High Availability Servers and Applications

For information on how to configure and tune applications and Communications Server for best performance with high availability, see the Performance Tuning Guide on docs.sun.com, 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 Communications ServerProvides High Availability

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

- "Storage for Session State Data" on page 21
- "In-Memory Replication on Other Servers in the Cluster" on page 21
- "Highly Available Clusters" on page 21

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. Communications 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 56.

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 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.
- 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.

Recovering from Failures

Using Sun Cluster

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

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:

- "Recovering the Domain Administration Server" on page 23
- "Recovering Node Agents and Server Instances" on page 23
- "Recovering Message Queue" on page 24

Recovering the Domain Administration Server

Loss of the Domain Administration Server (DAS) affects only administration. Communications 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 25.
- 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 Server installation 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 Communications Server with node agent on the machine.
2. Recreate the node agents. You do not need to create any server instances.
3. Synchronization will copy and update the configuration and data from the DAS.

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 Communications 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 Communications 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 Communications 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:

- "How the Converged Load Balancer Works" on page 29
- "Setting Up Converged Load Balancing" on page 34
- "Configuring the Converged Load Balancer" on page 37
- "The Data Centric Rules File" on page 41

This chapter discusses using the converged load balancer included with the Communications Server, which load balances HTTP, HTTPS, SIP, and SIPS messages.

Another load balancing option is to use the Sun Secure Application Switch with the Communications 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`).

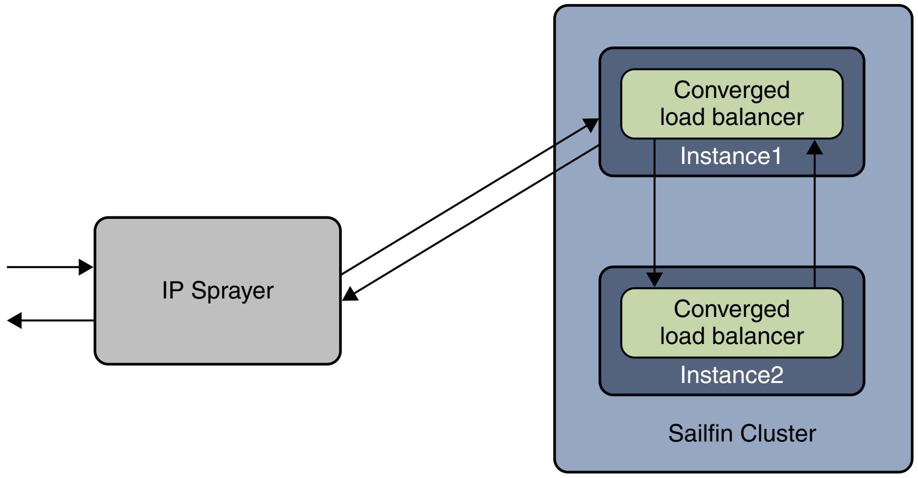
How the Converged Load Balancer Works

The load balancer attempts to evenly distribute the workload among multiple 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.

The following illustration depicts the working of a load balancer:



1. IP sprayer receives the client request.

   Note – The IP sprayer could be a hardware IP sprayer, which distributes requests evenly across all instances in a cluster. A network element such as the IP sprayer can front the converged load balancer and , at the transport level, distribute traffic over the cluster.

   ---

2. IP sprayer selects any of the SailFin instances in the cluster and forwards request to that instance. This example and illustration shows the request being forwarded to Instance1.
3. The converged load balancer on Instance1 selects an instance (Instance2, in this case) to service the request.

   The selection of the server instance to forward the request, is based on the configured "Converged Load Balancing Algorithms" on page 31. This key is added to the request as a header or parameter for maintaining stickiness.
4. In this example, the converged load balancer on Instance1 forwards the request to Instance2. If the converged load balancer chooses Instance1 to service this request, step 5 is bypassed.
5. The converged load balancer on Instance2 receives the request and detects that the request is already proxied from another instance. Without any further processing, Instance2 passes the request to the container so that it can be serviced. The converged load balancer on Instance2 sends the response back to the converged load balancer on Instance1.
6. The converged load balancer on Instance1 sends the response back to the client

7. Once a session is established, sticky information is set on the response and subsequent requests will carry this sticky information. Subsequent requests from the client will have the sticky information in the header/parameter. Even if Instance1 receives this request, it detects the sticky information and forwards the request to the converged load balancer on Instance2.

   To maintain HTTP and HTTPS session stickiness the load balancer uses cookies or , if the browser does not support cookies, uses URL rewriting. For SIP/SIPS sessions, the load balancer uses parameters such as the `BEKey`and `BERoute`. For example, the converged load balancer stamps the `BERoute` parameter on the VIA header as part of the outgoing request.

Converged Load Balancing Algorithms

The load balancer automatically uses one of the following algorithms:

- Round-robin algorithm — The load balancer selects the instance for servicing new message in a round robin fashion.
- Consistent hash — The load balancer selects the instance for servicing new message based on hash-key extracted from request. The hash key is extracted by using the rule specified in the DCR file, if provided. If a DCR file is not provided, hash key is extracted using default headers.

  A DCR file, `data-centric-rules.xml`, provides rules for applying consistent hashing on both HTTP/HTTPS and SIP/SIPS messages from converged or pure SIP applications. If this file is specified, the instructions in this file override the mechanism to extract hash key using default headers. If a DCR file is not provided, SIP and HTTP messages that are part of the same session may be serviced by different instances. Ensure that you provide a DCR file when you are deploying a converged application. The default header used may not provide a correct load distribution for SIP messages either. Therefore, even for pure sip applications, it is recommended that you provide a DCR file. For more information about this file, see "Editing Configuration Settings" on page 37 and "The Data Centric Rules File" on page 41.

  In addition to this XML, it is possible to configure DCR by using a plug-in in the form of a Java class.

Converged Load Balancing Algorithm for Web Applications

HTTP and HTTPS messages belonging to pure web applications, 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, it may result in creation of a session. In such a case, response will have sticky information, which is sent back on subsequent messages. Subsequent messages from the same client for the same session-based application are considered assigned or sticky messages 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 new requests.

Converged Load Balancing Algorithm for SIP Applications

For SIP and SIPS messages belonging to pure SIP applications, the converged load balancer uses a *consistent hash* algorithm, by default. If any of the rules in the DCR file match the SIP/SIPS request, a hash key is extracted using that rule. If none of the instructions or rules in the DCR file match the SIP or SIPS request, a hash key is generated using the `from-tag,call-id` parameters of the request.

Converged Load Balancing Algorithm for Converged Applications

Converged load balancer applies appropriate algorithms for HTTP/HTTPS and SIP/SIPS messages from converged applications, as follows:

- If no DCR file is specified, applies the *sticky round robin* algorithm for HTTP/HTTPS messages and the *consistent hash* algorithm for SIP/SIPS messages.
- If a DCR file is specified, applies the *consistent hash* algorithm for both HTTP/HTTPS messages and SIP/SIPS messages.

  For HTTP and HTTPS messages belonging to converged applications, if any of the rules in the DCR file match the HTTP/HTTPS request, a hash key is extracted using that rule. If none of the instructions or rules in the DCR file match the HTTP/HTTPS request, a hash key is extracted using remote host and port of the HTTP request.

  For SIP and SIPS messages, if any of the rules in the DCR file match the SIP/SIPS request, a hash key is extracted using that rule. If none of the instructions or rules in the DCR file match the SIP or SIPS request, a hash key is generated using the `from-tag,call-id` parameters of the request.

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 Self-Load-Balancing Clusters" on page 32
- "Using Dedicated Load Balancing Server Instances" on page 33

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.

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.

  All instances in a cluster must have similar view for each other's IP address. For example, consider a cluster with two instances, say instance1 and instance2. When instance1 looks up instance2's IP address, it gets a.b.c.d. When instance2 looks up its own IP address, it should also see a.b.c.d, which is the same IP viewed by instance1. Ensure that the hosts file on your machine is correctly set up.

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

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 Server clusters or server instances to participate in load balancing using the cluster profile. For more information, see Chapter 3, "Setting Up Clusters in Communications Server."
- Make sure the group management service (GMS) is enabled. It is enabled by default. For more information, see "Group Management Service" on page 56.
- Create node agents for these clusters or server instances. For more information, see Chapter 4, "Configuring Node Agents."
- Configure session persistence using in-memory replication. For more information, see Chapter 6, "Configuring High Availability Session Persistence and Failover"
- Deploy applications to these clusters or instances. For more information, see the *Sun GlassFish Communications Server 2.0 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*`/config/`*cluster-config*`/converged-loadbalancer.xml`.
- Automatically Apply Changes — Specifies whether to automatically apply configuration changes to the target server instances. This setting is off by default. Set this to true and generate the converged load balancer's configuration file.
- 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 `asadmin` Commands

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.

Note – While you perform the iterative process of setting up and defining the cluster deployment, it is recommended that `--autocommit` be set to false. By default, this option is set to false, primarily to prevent generation of intermediate converged load balancer files, which would get generated for every change in `domain.xml`, which impact the converged load balancer's view of its configuration. After the cluster deployment definition has reached a stable point in evolution, set `--autocommit` to true.

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 --user admin --passwordfile pass.txt
--host host1 nodeagent1
    Command create-node-agent executed successfully.
asadmin>create-node-agent --user admin --passwordfile pass.txt
--host host1 nodeagent2
    Command create-node-agent executed successfully.
asadmin>create-instance --user admin --passwordfile pass.txt --nodeagent nodeagent1
--cluster cluster1 cluster1_instance1
   Command create-instance executed successfully.
asadmin> create-instance --user admin --passwordfile pass.txt
--nodeagent nodeagent2 --cluster cluster1 cluster1_instance2
   Command create-instance executed successfully.
asadmin> create-converged-lb --user admin --passwordfile pass.txt
--configfile clb.xml --autocommit=true --lbenableallinstances=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 *Sun GlassFish Communications Server 2.0 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 37
- "Editing Load Balancer Settings" on page 39
- "Enabling or Disabling a Server Instance for Load Balancing" on page 40
- "Changing the Log Message Level for the Converged Load Balancer" on page 41

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 37
- "Editing Load Balancer Details" on page 38
- "Editing Self Load Balancing (Load Balancer Target Details)" on page 39

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,call-id`.

TABLE 2–1 Load Balancer Configuration Settings *(Continued)*

Setting	Description
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 41. 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,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 Settings

Setting	Description
Automatically Apply Changes	Specifies whether to automatically apply configuration changes to the target server instances.
Configuration File Name	Specifies the name of the converged load balancer's configuration file. The default path and name of the configuration file is at *domain-dir*/config/*cluster-config*, where *cluster-config*is the configuration repository directory specific to the cluster's configuration. By default, the name of the configuration file name is *clb-name*`_CLB_CONFIG.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 (Load Balancer Target Details)

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

To edit Load Balancer 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

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-name-config.availability-service.converged-load-balancer.setting
```

For example:

```
asadmin set c1-config.availability-service.converged-load-balancer.config-file=myclb.xml
```

```
asadmin set c1-config.availability-service.converged-load-balancer.auto-commit=true
```

```
asadmin set c1-config.availability-service.converged-load-balancer.converged-lb-config-name=myclb-config
```

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

```
asadmin set config-name-config.availability-service.converged-load-balancer.proxy.setting
```

For example:

```
asadmin set c1-config.availability-service.converged-load-balancer.proxy.request-pool-size=50
asadmin set c1-config.availability-service.converged-load-balancer.proxy.send-retry-count=3
asadmin set c1-config.availability-service.converged-load-balancer.proxy.read-time-out=1500
```

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

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

For example:

```
asadmin set domain.converged-lb-configs.myclb-config.converged-lb-policy.http=round-robin

asadmin set domain.converged-lb-configs.myclb-config.converged-lb-policy.sip=from-tag,call-id
```

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 domain.converged-lb-configs.myclb-config.converged-lb-cluster-ref.clust1.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 *Sun GlassFish Communications Server 2.0 Reference Manual*.

Enabling or 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 or cluster for load balancing, use the `asadmin disable-converged-lb-server` command.

For example:

```
asadmin disable-converged-lb-server --user admin --passwordfile pass.txt cluster1
```

You can use the `asadmin enable-converged-lb-server` command to enable a server instance or cluster for load balancing. A new instance or cluster is created with the `lb-enabled` option set to false by default. You need to explicitly enable an instance or cluster for load balancing.

For example:

```
asadmin enable-converged-lb-server --user admin --passwordfile pass.txt cluster1
```

For more information about the `asadmin disable-converged-lb-server` command and the `asadmin enable-converged-lb-server`, see the *Sun GlassFish Communications Server 2.0 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 `c1-config` to `FINE` as follows:

```
asadmin set c1-config.log-service.module-log-levels.property.clb=FINE
```

```
asadmin set c1-config.log-service.module-log-levels.property.sip=FINE
```

The Data Centric Rules File

A consistent hash algorithm determines the server instance to which the request is forwarded. The server instance selection is based on a hash key. You can define data centric rules, using which the key is extracted from incoming SIP and HTTP requests. Data centric rules are not application-specific and apply to all applications deployed on the LB targets. Only new initial requests are affected by such changes.

The default path to the data centric rules file is *domain-dir*`/config/`*cluster-config-name*. 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 is *domain-dir*`/config/`.

Data centric rules are dynamically re-configurable. The admin framework generates an admin event into the converged load balancer where the data centric rules file is supplied as an argument and the converged load balancer loads a new version of the data centric rules file.

If the data centric rules do not match any incoming SIP or HTTP requests, one of the following happens: .

- For HTTP requests, a hash key is generated using the remote host and port.
- For SIP requests, a hash key is generated using `from-tag,call-id`

The data centric rules file can be a JAR file or an XML file, as described in the following sections:

- "Creating a Data Centric Rules JAR File" on page 41
- "Creating a Data Centric Rules XML File" on page 44

  To set a data centric rules JAR or XML file for a converged load balancer, see the instructions at "Setting the Data Centric Rules File for a Converged Load Balancer" on page 53

Creating a Data Centric Rules JAR File

The data centric rules of the converged load balancer is configurable using an implementation of the `org.glassfish.comms.api.datacentric.DcrPlugin` interface , which is supplied as a pre-compiled Java class packaged in a JAR file. When routing a request, the converged load balancer calls the instantiated plug-in to extract the key from initial SIP and HTTP requests. The converged load balancer uses the result returned from the plug-in, to lookup the serving instance in the consistent hash.

The `org.glassfish.comms.api.datacentric.DcrPlugin` interface specifies two methods:

- `String getKey(javax.servlet.sip.SipServletRequest request, javax.servlet.sip.SipFactory sipFactory, org.glassfish.comms.api.uriutils.UriTools uriTools, org.glassfish.comms.api.telurl.TelUrlResolver telUrlResolver)`
- `String getKey(javax.servlet.http.HttpServletRequest request, javax.servlet.sip.SipFactory sipFactory, org.glassfish.comms.api.uriutils.UriTools uriTools, org.glassfish.comms.api.telurl.TelUrlResolver telUrlResolver)`

You can implement these methods to extract the key to be used for lookup in the consistent hash in the converged load balancer, for initial requests. Each method can return a non-null value, which indicates that the request matches a rule and a hash key is successfully extracted. If none of the rules match a request, a null value is returned and the hash key is extracted using default rules.

To facilitate the implementation of the data centric rules plug-in, the following entities are provided as method arguments:

- `TelUrlResolver` — For resolving Tel URIs
- `UriTools`— For normalizing and canonicalizing URIs
- `SipFactory` — For converting header values to URIs and extracting the various components from such a URI

The data centric rules JAR file must contain a compiled implementation of the plug-in interface. It can also contain other application specific helper classes. It must also have a manifest that specifies the class name of this data centric rules plug-in implementation. The converged load balancer uses the manifest to identify the class that implements data centric rules plug-in. It then loads the plug-in class and creates an instance.

Here is an example Java class file that specifies data centric rules:

```
package mydcr;

import org.glassfish.comms.api.datacentric.DcrPlugin;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.sip.SipFactory;
import javax.servlet.sip.SipServletRequest;
import javax.servlet.sip.SipURI;
import org.glassfish.comms.api.telurl.TelUrlResolver;
import org.glassfish.comms.api.uriutils.UriTools;

public class MyDcrPluginImpl implements DcrPlugin{

    public String getKey(SipServletRequest request,
            SipFactory sipFactory, UriTools uriTools,
            TelUrlResolver telUrlResolver) {
        String key = null;
        String method = request.getMethod().toUpperCase();
        if(method.equals( "PUBLISH" ) || method.equals( "INVITE" )
              || method.equals( "CANCEL" )){
           key = SipURI.class.cast(request.getTo().getURI()).getUser();
        } else if (method.equals( "REGISTER" )
                || method.equals( "SUBSCRIBE" )){
            key = SipURI.class.cast(request.getFrom().getURI()).getUser();
        }
        System.out.println("MY DCR PLUGIN - Key extracted from SIP request : " + key);
        return key;
    }

    public String getKey(HttpServletRequest request,
            SipFactory sipFactory, UriTools uriTools,
            TelUrlResolver telUrlResolver) {
        String key =  request.getParameter( "j_username" );
        System.out.println("MY DCR PLUGIN - Key extracted from HTTP request : " + key);
        return key;
    }


}
```

After you have written the data centric rules class, follow these steps:

1. Compile the data centric rules class as shown in the following example command. You must include `ssa-api.jar`, `javaee.jar`, and `comms-appserv-api.jar` in the classpath. The `classes` directory must exist.

```
javac -classpath as-install/lib/ssa-api.jar:as-install/lib/javaee.jar:as-install/lib/comms-appserv-api.jar
-d classes src/mydcr/MyDcrPluginImpl.java
```

2. Write the manifest file. For example:

   ```
   Manifest-Version: 1.0
   Dcr-Plugin-Class: mydcr.MyDcrPluginImpl
   ```

3. Create the data centric rules JAR file. For example:

   ```
   jar -cvfm mydcrplugin.jar mymanifest -C classes mydcr/MyDcrPluginImpl.class
   ```

4. Set the data centric rules file using the `asadmin set-dcr-file` command. Refer the instructions at "Setting the Data Centric Rules File for a Converged Load Balancer" on page 53.

Creating a Data Centric Rules XML File

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.

Here is an example XML file that specifies data centric rules:

```
<?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 default name is `data-centric-rules.xml`. The DTD file that validates the file format is *as-install*`/lib/dtds/sun-data-centric-rule_1_0.dtd`.

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

- "Top-Level Elements" on page 45
- "Operator Elements" on page 46
- "Condition Elements" on page 48
- "Condition Type Elements" on page 50
- "Variables" on page 51

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 46, "http-rules" on page 46.

sip-rules

Determines the data centric rules for SIP and SIPS requests.

Superelements

"user-centric-rules" on page 45

Subelements

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

"or" on page 46, "and" on page 47, "if" on page 47, "header" on page 48, "request-uri" on page 48, "session-case" on page 49, "cookie" on page 49

http-rules

Determines the data centric rules for HTTP and HTTPS requests.

Superelements

"user-centric-rules" on page 45

Subelements

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

"or" on page 46, "and" on page 47, "if" on page 47, "header" on page 48, "request-uri" on page 48, "session-case" on page 49, "cookie" on page 49

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 48.

or

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

Superelements

"sip-rules" on page 46, "http-rules" on page 46, "or" on page 46, "and" on page 47, "if" on page 47

Subelements

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

"or" on page 46, "and" on page 47, "if" on page 47, "header" on page 48, "request-uri" on page 48, "session-case" on page 49, "cookie" on page 49

and

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

Superelements

"sip-rules" on page 46, "http-rules" on page 46, "or" on page 46, "and" on page 47, "if" on page 47

Subelements

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

"or" on page 46, "and" on page 47, "if" on page 47, "header" on page 48, "request-uri" on page 48, "session-case" on page 49, "cookie" on page 49

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 46, "http-rules" on page 46, "or" on page 46, "and" on page 47, "if" on page 47

Subelements

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

"or" on page 46, "and" on page 47, "if" on page 47, "header" on page 48, "request-uri" on page 48, "session-case" on page 49, "cookie" on page 49

The "else" on page 47 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 47

Attributes

The following attribute is required and case-sensitive.

`return`	Specifies a variable that evaluates to the return value. See "Variables" on page 51.

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 46, "http-rules" on page 46, "or" on page 46, "and" on page 47, "if" on page 47

Subelements

Exactly one of the following subelements is required:

"exist" on page 50, "notexist" on page 50

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 51.

request-uri

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

Superelements

"sip-rules" on page 46, "http-rules" on page 46, "or" on page 46, "and" on page 47, "if" on page 47

Subelements

Exactly one of the following subelements is required:

"exist" on page 50, "notexist" on page 50, "match" on page 51

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 51.

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 50 subelement description.

Superelements

"sip-rules" on page 46, "http-rules" on page 46, "or" on page 46, "and" on page 47, "if" on page 47

Subelements

The "equal" on page 50 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 46, "and" on page 47, "if" on page 47, "header" on page 48, "request-uri" on page 48, "session-case" on page 49, "cookie" on page 49

cookie

Specifies a rule based on an HTTP cookie.

Superelements

"sip-rules" on page 46, "http-rules" on page 46, "or" on page 46, "and" on page 47, "if" on page 47

Subelements

Exactly one of the following subelements is required:

"exist" on page 50, "notexist" on page 50

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 51.

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 51.

Superelements

"header" on page 48, "request-uri" on page 48, "cookie" on page 49

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 51.

Superelements

"header" on page 48, "request-uri" on page 48, "cookie" on page 49

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 49

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.

Superelements

"request-uri" on page 48

Variables

The `name`, `parameter`, and `return` attributes and the "exist" on page 50 and "notexist" on page 50 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,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

Setting the Data Centric Rules File for a Converged Load Balancer

The following tasks use the Admin Console or the CLI to set a data centric rules file for a converged load balancer.

▼ To Set the Data Centric Rules File Using the Admin Console

1 **Create the data centric rules file.**

2 **Log in to the Admin Console.**

In you browser, type `http://`*hostname*`:`*port*

3 **In the left-hand pane, click Converged Load Balancer.**

4 **Select a Converged Load Balancer from the list of Converged Load Balancers.**

The Settings page appears. The DCR File Settings section in this page displays the current data centric rules file.

5 **In the Upload DCR File field, click Browse to upload the data centric rules XML or JAR file you have created.**

Note – Communications Server does not expect the file extension to be case-sensitive.

---

6 **Click Save.**

▼ To Set the Data Centric Rules File Using CLI

1 **Create the data centric rules file.**

2 **To set the XML or JAR file as the data centric rules file for Communications Server, use the `asadmin set-dcr-file` command.**

The following are sample commands:

```
asadmin set-dcr-file --host myhost --port 4850 --clbconfig my-cluster-config
mydcrfile.jar
```

```
asadmin set-dcr-file --host myhost --port 4850 --clbconfig my-cluster-config
mydcrfile.xml
```

Load Balancers with Enterprise Server and Communications Server

The converged load balancer is part of the Communications Server distribution. The HTTP load balancing plug-in is bundled with Sun GlassFish Enterprise Server with HADB bundle. The HTTP load balancing plug-in is also available as a separate download from `http://glassfish.dev.java.net`

HTTP Load Balancer and Converged Load Balancer

For pure web application enterprise deployments , use the HTTP load balancing plug-in with Sun GlassFish Enterprise Server 2.1. Sun GlassFish Communication Server with the Converged Load Balancer is targeted at enterprise deployments of SIP and converged applications.

◆ ◆ ◆ CHAPTER 3

Setting Up Clusters in Communications Server

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

- "Overview of Clusters" on page 55
- "Group Management Service" on page 56
- "Working with Clusters" on page 60
- "Using the Multi-homing Feature With a Cluster" on page 68

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.

Instances can be grouped into clusters. You can distribute an application to all instances in the cluster with a single deployment. Clusters are dynamic. When an instance is added or removed, the changes are handled automatically.

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.

Cluster instances are organized in a ring topology. Each member in the ring sends in-memory state data to the next member in the ring, its replica partner, and receives state data from the previous member. As state data is updated in any member, it is replicated around the ring. When a member fails in the ring topology, the ring is broken. Group Management Service (GMS) can recognize the failure of a member. In that event, the replication framework reshapes the topology of the cluster and notifies members of the changes. When a member learns that its replica partner has disappeared, it selects a new partner from in-service members.

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 Communications 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/`).

The following topics are addressed here:

- "GMS Failure Detection Settings" on page 56
- "To Enable or Disable GMS for a Cluster" on page 58
- "Configuring GMS" on page 58

GMS Failure Detection Settings

The following settings are used in GMS failure detection:

`fd-protocol-max-tries`
: Indicates the maximum number of missed heartbeats that the health monitor counts before marking an instance as suspected failure. GMS also tries to make a peer-2-peer connection with the suspected member. If that also fails, the member is marked as suspect failed.

`fd-protocol-timeout-in-millis`
: Indicates the failure detection interval (in milliseconds) between each heartbeat message that would provoke an instance to send out its Alive message. This setting considers the number of milliseconds between missed heartbeats that the `max-retry` logic would wait for, in the master node, between counting each missed heartbeat. Lowering the value of retries would mean that failure would be suspected after fewer missed heartbeats. Lowering the value of `fd-protocol-timeout-in-millis` below the default would result in more frequent heartbeat messages being sent out from each member. This could potentially result in more

heartbeat messages in the network than a system needs for triggering failure detection protocols. The effect of this varies depending on how quickly the deployment environment needs to have failure detection performed. That is, the (lower) number of retries with a lower heartbeat interval would make it quicker to detect failures. However, lowering the timeout or retry attempts could result in false positives because you could potentially detect a member as failed when, in fact, the member's heartbeat is reflecting the network load from other parts of the server. Conversely, a higher timeout interval results in fewer heartbeats in the system because the time interval between heartbeats is longer. As a result, failure detection would take a longer. In addition, a startup by a failed member during this time results in a new join notification but no failure notification, because failure detection and evaluation were not completed. The lack of a join notification without a preceding failure notification is logged.

`ping-protocol`
Indicates the amount of time an instance's GMS module will wait during instance startup (on a background thread, so that server startup does not wait for the timeout) for discovering the master member of the group. In GMS, this process is called master node discovery protocol. The instance's GMS module sends out a master node query to the multicast group address. If the instance times out (does not receive a master node response from another member within this time) the master is assumed absent and the instance assumes the master role. The instance sends out a master node announcement to the group, and starts responding to subsequent master node query messages from members. In Communications Server, the domain administration server (DAS) joins a cluster as soon as it is created, which means the DAS becomes a master member of the group. This allows cluster members to discover the master quickly, without incurring a timeout. Lowering the `ping-protocol` timeout would cause a member to timeout more quickly because it will take longer to discover the master node. As a result, there might be multiple masters in the group which could lead to master collision. Master collision could cause resolution protocol to start. The master collision, and resolution protocol, results in multiple masters telling each other who the true master candidate is based on sorted order of memberships (based on their UUIDs). The impact can be extensive in messaging if there are many masters in the group. Therefore, the `ping-protocol` timeout value should be set to the default or higher.

`vs-protocol-timeout-in-millis`
Indicates the verify suspect protocol's timeout used by the health monitor. After a member is marked as suspect based on missed heartbeats and a failed peer–2–peer connection check, the verify suspect protocol is activated and waits for the specified timeout to check for any further health state messages received in that time, and to see if a peer-2-peer connection can be made with the suspect member. If not, then the member is marked as failed and a failure notification is sent.

The retries, missed heartbeat intervals, peer-2-peer connection-based failure detection, watchdog-based failure reporting, and the verify suspect protocols are all needed to ensure that failure detection is robust and reliable in Communications Server.

▼ 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.

Sample `get` command to get all the properties associated with a *cluster-config-name*.

```
asadmin get cluster2-config.group-management-service.*

cluster2-config.group-management-service.fd-protocol-max-tries = 3
cluster2-config.group-management-service.fd-protocol-timeout-in-millis = 2000

cluster2-config.group-management-service.merge-protocol-max-interval-in-millis
= 10000

cluster2-config.group-management-service.merge-protocol-min-interval-in-millis
= 5000

cluster2-config.group-management-service.ping-protocol-timeout-in-millis = 5000

cluster2-config.group-management-service.vs-protocol-timeout-in-millis = 1500
```

▼ To Configure GMS Settings Using Admin Console

1 In the Admin Console, go to Communications Server node

2 Click Configuration –> *cluster_name*`-config` **—> Group Management Service.**

Example 3–1 Changing GMS Settings Using `asadmin` `get` and `set`commands

Instead of using the Admin Console, you can use the `asadmin` `get` and `set` commands.

```
asadmin> list  cluster2-config.*
cluster2-config.admin-service
cluster2-config.admin-service.das-config
cluster2-config.admin-service.jmx-connector.system
cluster2-config.admin-service.jmx-connector.system.ssl
cluster2-config.availability-service
cluster2-config.availability-service.jms-availability
cluster2-config.availability-service.sip-container-availability
cluster2-config.diagnostic-service
cluster2-config.ejb-container
cluster2-config.ejb-container-availability
cluster2-config.ejb-container.ejb-timer-service
...
...
...
...
cluster2-config.web-container-availability

asadmin> get  cluster2-config.group-management-service.*
cluster2-config.group-management-service.fd-protocol-max-tries = 3
cluster2-config.group-management-service.fd-protocol-timeout-in-millis = 2000
cluster2-config.group-management-service.merge-protocol-max-interval-in-millis = 10000
cluster2-config.group-management-service.merge-protocol-min-interval-in-millis = 5000
cluster2-config.group-management-service.ping-protocol-timeout-in-millis = 5000
cluster2-config.group-management-service.vs-protocol-timeout-in-millis = 1500

asadmin>set  cluster2-config.group-management-service.fd-protocol-max-tries=4
 cluster2-config.group-management-service.fd-protocol-max-tries = 4

asadmin> get  cluster2-config.group-management-service.*
 cluster2-config.group-management-service.fd-protocol-max-tries = 4
cluster2-config.group-management-service.fd-protocol-timeout-in-millis = 2000
cluster2-config.group-management-service.merge-protocol-max-interval-in-millis = 10000
cluster2-config.group-management-service.merge-protocol-min-interval-in-millis = 5000
cluster2-config.group-management-service.ping-protocol-timeout-in-millis = 5000
cluster2-config.group-management-service.vs-protocol-timeout-in-millis = 1500
```

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

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.

Server instances can reside on different machines. Every server instance needs to be associated with a node agent that can communicate with the DAS. 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 84

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 62
- "To Create Server Instances for a Cluster" on page 61
- "To Configure Applications for a Cluster" on page 64
- "To Configure Resources for a Cluster" on page 64
- "To Delete a Cluster" on page 65

For more details on how to administer clusters, server instances, and node agents, see "Deploying Node Agents" on page 75.

▼ 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 84

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 73
- “To Create a Cluster” on page 60
- “To Configure a Cluster” on page 62
- “To Configure Applications for a Cluster” on page 64
- “To Configure Resources for a Cluster” on page 64
- “To Delete a Cluster” on page 65
- “To Configure Server Instances in a Cluster” on page 63

▼ 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 60
- “To Create Server Instances for a Cluster” on page 61
- “To Configure Applications for a Cluster” on page 64
- “To Configure Resources for a Cluster” on page 64
- “To Delete a Cluster” on page 65

- "To Migrate EJB Timers" on page 65

▼ 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 60
- "To Configure a Cluster" on page 62
- "To Create Server Instances for a Cluster" on page 61

- "To Configure Applications for a Cluster" on page 64
- "To Configure Resources for a Cluster" on page 64
- "To Delete a Cluster" on page 65
- "To configure how the Communications Server recovers from transactions" in *Sun GlassFish Communications Server 2.0 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 60
- "To Configure a Cluster" on page 62
- "To Create Server Instances for a Cluster" on page 61
- "To Configure Resources for a Cluster" on page 64
- "To Delete a Cluster" on page 65

▼ 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 Disable. 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 60
- "To Configure a Cluster" on page 62
- "To Create Server Instances for a Cluster" on page 61
- "To Configure Applications for a Cluster" on page 64
- "To Delete a Cluster" on page 65

▼ 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 60
- "To Configure a Cluster" on page 62
- "To Create Server Instances for a Cluster" on page 61
- "To Configure Applications for a Cluster" on page 64
- "To Configure Resources for a Cluster" on page 64

▼ 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 62
- Admin Console online help for configuring settings for the EJB timer service

▼ To Upgrade Components Without Loss of Service

In a clustered environment, a rolling upgrade redeploys an application with a minimal loss of service and sessions. A session can be any replicable artifact, including:

- `HttpSession`
- `SingleSignOn`
- `SipApplicationSession`
- `SipSession`
- `ServletTimer`
- `DialogFragment`
- stateful session bean

You can use the load balancer and multiple clusters to upgrade components within the Communications Server without any loss of service. A component can, for example, be a JVM, the Communications Server, or a web application.

A rolling upgrade can take place under light to moderate load conditions. The procedure should be doable in a brief amount of time, about 10-15 minutes per server instance.

Applications must be compatible across the upgrade. They must work correctly during the transition, when some server instances are running the old version and others the new one. The old and new versions must have the same shape (for example, non-transient instance variables) of `Serializable` classes that form object graphs stored in sessions. Or, if the shape of these classes is changed, then the application developer must ensure that correct `Serialization` behavior occurs. If the application is not compatible across the upgrade, the cluster must be stopped for a full redeployment.

The `Basic3pcc` sample application includes an Ant target, `do-rollingupgrade`, which performs all the rolling upgrade steps for you. This sample application is included with the Communications Server in the *as-install*`/samples/sipservlet/Basic3pcc` directory. The Basic3pcc application and the Ant target are available only with the JAR installer of Communications Server.

The following procedure describes how to upgrade an application running on all instances of a cluster.

1 **Run the following commands on the converged load balancer in the cluster,**

```
asadmin set
domain.converged-lb-configs.clb_config_name.property.load-increase-factor=1
```

```
asadmin set
domain.converged-lb-configs.clb_config_name.property.load-factor-increase-period-in-se
```

2 **Set the value of the `dynamic-reconfig` attribute to false in the cluster.**

3 **Redeploy a new version of the application.**

Because you have set the `dynamic-reconfig` attribute to false, the new version of the application will be loaded to the instance only when the instance restarts.

4 **Disable the instance from the converged load balancer by using the following asadmin command:**

```
asadmin disable-converged-lb-server instance_name
```

5 **Back up the current session with the following command:**

```
asadmin backup—session store instance_name
```

By default, the session files are stored at *instance-dir*/rollingupgrade.

6 **Stop the instance with the following command:**

```
asadmin stop-instanceinstance_name
```

7 **Start the instance.**

```
asadmin start-instance instance_name
```

8 **Restore the session.**

```
asadmin restore–session–store instance_name
```

9 **Enable the instance to the converged load balancer.**

```
asadmin enable-converged-lb-server instance_name
```

10 **Use the following command to get the latest version of the session store, which could have been updated by another instance accessing this session store.**

```
asadmin reconcile–session–store instance_name
```

11 **For all instances in the cluster, repeat steps 3 to 9.**

12 **Set the value of the `dynamic-reconfig` attribute to true in the cluster.**

Using the Multi-homing Feature With a Cluster

Multi-homing enables Communication Server clusters to be used in an environment that uses multiple Network Interface Cards (NICs). A multi-homed host has multiple network connections, of which the connections may or may not be the same network. Multi-homing provides the following benefits:

- Provides redundant network connections within the same subnet. Having multiple NICs ensures that one or more network connections are available for communication.
- Supports SIP communication across two or more different subnets. For example, for proxying SIP requests from User Agents in one subnet to User Agents in a second subnet, when the User Agents cannot directly communicate across subnets.
- Binds to a specific IPv4 or IPv6 address and receives SIP messages from thatip:port in a system that has multiple IP addresses configured. The responses for SIP requests received on a particular interface will also go out through that interface.
- Binds to a specific IPv4 or IPv6 address and receives SIP and HTTP messages from thatip:port in a system that has multiple IP addresses configured. The responses for SIP requests received on a particular interface will also go out through that interface.
- Allows for configuring more than one external and/or more than one internal SIP listener. Configuring more than one internal listener would mean that the converged load balancer would use these implicitly for proxying in a round-robin mechanism.
- Supports separation of external and internal traffic.

Traffic Separation Using Multi-homing

You can separate the internal traffic (resulting from the converged load balancer, replication and GMS) from the external traffic. Traffic separation enables you plan a network better and augment certain parts of the network, as required.

Consider a simple cluster, `cluster1`, with three instances, `instance101`, `instance102`, and `instance103`. Each instance runs on a different machine. In order to separate the traffic, the multi-homed machine should have at least two IP addresses belonging to different networks. The first IP as the external IP and the second one as internal IP. The objective is to expose the external IP to the User Agents, so that all the traffic from the User Agents would be through them. The internal IP is used only by the cluster instances for internal communication. The following procedure describes how to set up traffic separation.

▼ To Set Up Traffic Separation

1 **Set the address attribute of SIP listeners and HTTP listeners to the external address of the mutli-homed machine.**

Use the following commands:

- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1—config.sip-service.sip-listener.sip-listener1.address=\${EXTERNAL_LISTENE`
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1—config.http-service.http-listener.http-listener1.address=\${EXTERNAL_LIST`
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1—config.sip-service.sip-listener.sip-listener2.address=\${EXTERNAL_LISTENE`
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1—config.sip-service.http-listener.http-listener2.address=\${EXTERNAL_LISTE`

2 **Set the listener type of these listeners as external, so that they listen for traffic from User Agents and not for the converged load balancer proxying.**

- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1—config.sip-service.sip-listener.sip-listener1.type=external`
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1—config.http-service.http-listener.http-listener1.type=external`
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1—config.sip-service.sip-listener.sip-listener2.type=external`
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1—config.sip-service.http-listener.http-listener2.type=external`

3 **Create the system properties `EXTERNAL_LISTENER_ADDRESS` and `INTERNAL_LISTENER_ADDRESS`.**

- `asadmin create-system-properties --user admin --port 4848 --passwordfile password.txt --target cluster1 EXTERNAL_LISTENER_ADDRESS=0.0.0.0:INTERNAL_LISTENER_ADDRESS=0.0.0.0`
- `asadmin create-system-properties --user admin --port 4848 --passwordfile password.txt --target server EXTERNAL_LISTENER_ADDRESS=0.0.0.0:INTERNAL_LISTENER_ADDRESS=0.0.0.0`

4 Create new listeners for listening to internal traffic.

- `asadmin create-sip-listener --user admin --port 4848 --passwordfile password.txt --target cluster1 --siplisteneraddress 0.0.0.0 --siplistenerport 25060 internal-sip-listener`
- `asadmin create-http-listener --user admin --port 4848 --passwordfile password.txt --target cluster1 --listeneraddress 0.0.0.0 --defaultvs server --listenerport 28080 internal-http-listener`

5 Set the address attribute of these new listeners to the internal address.

- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1–config.sip-service.sip-listener.internal-sip-listener.address=\${INTERNAL_LIST`
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1–config.http-service.http-listener.internal-http-listener.address=\${INTERNAL_L`

6 Set the type attribute of these new listeners to internal.

- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1–config.sip-service.sip-listener.internal-sip-listener.type=internal`
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1–config.http-service.http-listener.internal-http-listener.type=internal`

7 Configure the IP address of the cluster instances.

- `asadmin create-system-properties --user admin --port 4848 --passwordfile password.txt --target instance101 EXTERNAL_LISTENER_ADDRESS=10.12.152.29:INTERNAL_LISTENER_ADDRESS=192.168.2.1`
- `asadmin create-system-properties --user admin --port 4848 --passwordfile password.txt --target instance102 EXTERNAL_LISTENER_ADDRESS=10.12.152.39:INTERNAL_LISTENER_ADDRESS=192.168.2.3`
- `asadmin create-system-properties --user admin --port 4848 --passwordfile password.txt --target instance103 EXTERNAL_LISTENER_ADDRESS=10.12.152.49:INTERNAL_LISTENER_ADDRESS=192.168.2.4`

8 Restart the node agent and the cluster.

9 **If you are using a hardware load balancer for spraying the SIP traffic to the individual instances, you need to set the `external-sip-address` and `external-sip-port` attributes to point to the hardware load balancer.**

If you are using only one hardware load balancer for all SIP listeners, set the attributes of the SIP container.

- `asadmin create-system-properties --user admin --port 4848 --passwordfile password.txt cluster1-config.sip-container.external-sip-address=`*yourlbaddress*
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1-config.sip-container.external-sip-port=`*yourlbport*

If you are using multiple hardware load balancers, set the attributes of each of the SIP listeners:

- `asadmin create-system-properties --user admin --port 4848 --passwordfile password.txt cluster1-config.sip-service.sip-listener.sip-listener1.external-sip-address=`*yourlba*
- `asadmin set --user admin --port 4848 --passwordfile password.txt cluster1-config.sip-service.sip-listener.sip-listener1.external-sip-port=`*yourlbport*

◆ ◆ ◆ CHAPTER 4

Configuring Node Agents

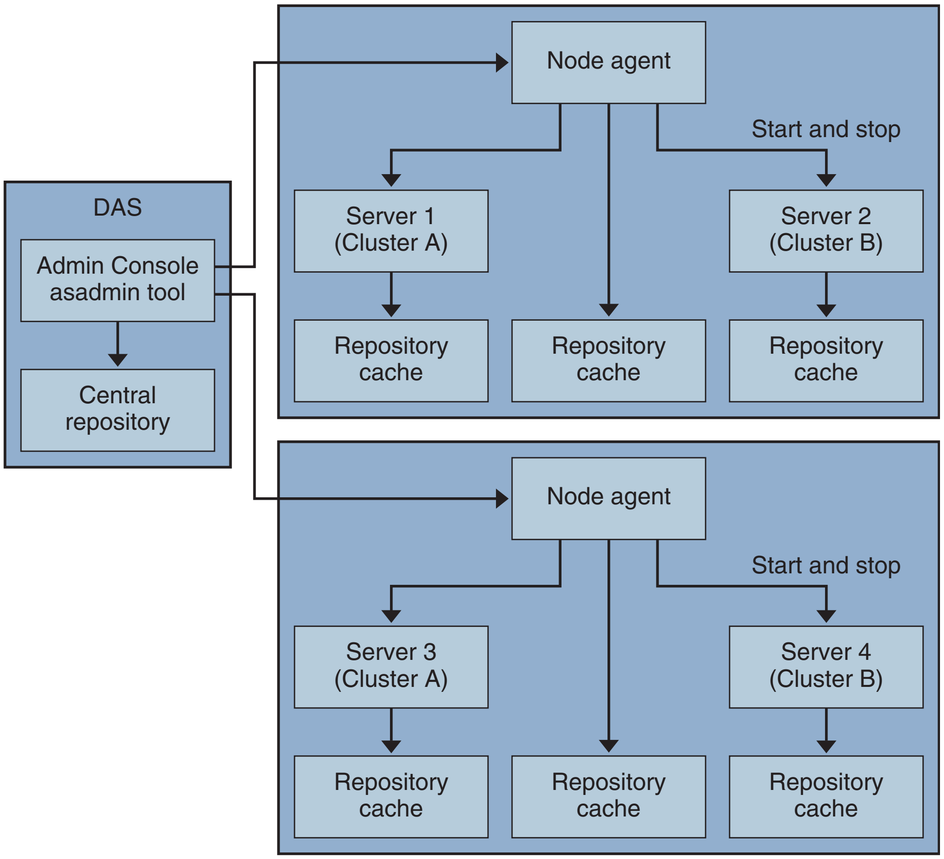
This chapter describes the node agents in Communications Server. It contains the following sections:

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 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

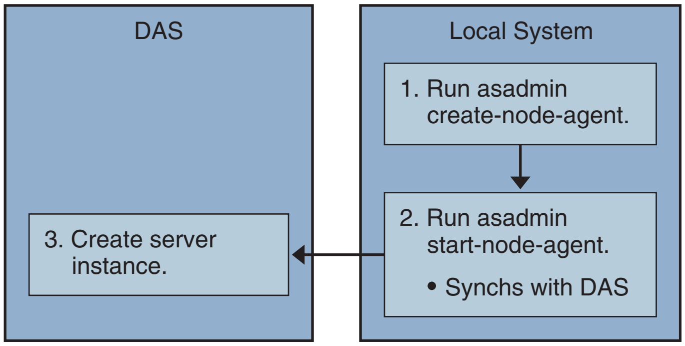
You configure and deploy node agents in two ways:

- *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 85 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.

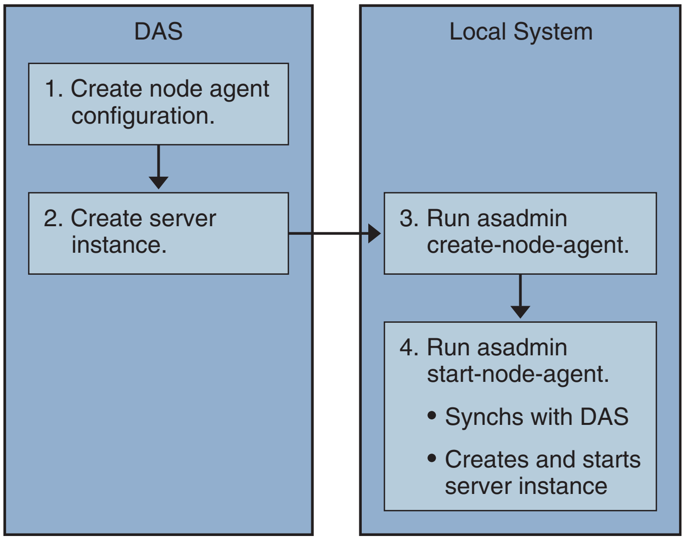
   See "Starting a Node Agent" on page 86 for more information.

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 84 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 85 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 86 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 77
- "Server Instance Synchronization" on page 78
- "Synchronizing Library Files" on page 80
- "Unique Settings and Configuration Management" on page 80
- "Synchronizing Large Applications" on page 81

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 4–1 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 GlassFish Communications Server 2.0 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>
```

Setting Timeouts for Starting and Stopping Node Agents

The following properties are available in Communications Server for setting the timeouts for starting and stopping the node agents: `com.sun.aas.startAgentTimeout` and `com.sun.aas.startAgentTimeout`.

These properties can be added to the *as-install*/`lib/processLauncher.xml` file. These properties are defined in milliseconds and they provide graceful starting and stopping of node agents when the `--startinstances` option is enabled.

If the node agent name is `mynodegant1`, find the `<process name="mynodegant1">` entry in the `/processLauncher.xml` file and add the following lines:

```
<sysproperty key="com.sun.aas.startAgentTimeout" value="1500000"/>
```

```
<sysproperty key="com.sun.aas.stopAgentTimeout" value="240000"/>
```

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

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 4–2 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 84

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 85 and "Starting a Node Agent" on page 86

▼ 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 85.

For a list of the steps involved in creating and using node agents, see "Deploying Node Agents" on page 75.

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 86.

▼ 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 4–1 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 86

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 85 and "Starting a Node Agent" on page 86.

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 87.

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 87 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 85.

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.

▼ To Create a Standalone Server Instance

1 In the tree component, select the Stand—Alone Instances node.

2 Click New.

3. **On the New Stand-Alone Server Instance page, provide the name of the instance.**

4. **Choose the node agent on which this instances is to be created and click OK.**

CHAPTER 5

Managing Configurations

This chapter describes adding, changing, and using named server configurations in Communications 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 94
- "Configurations Created when Creating Instances or Clusters" on page 94
- "Unique Port Numbers and Configurations" on page 95
- "To Create a Named Configuration" on page 96
- "Editing a Named Configuration's Properties" on page 96

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 94
- "Configurations" on page 93
- "To Create a Named Configuration" on page 96
- "Editing a Named Configuration's Properties" on page 96
- "To Edit Port Numbers for Instances Referencing a Configuration" on page 98

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 94
- "Configurations" on page 93
- "To Create a Named Configuration" on page 96
- "Editing a Named Configuration's Properties" on page 96

Clustered Configuration Synchronization

When you create a clustered configuration, Communications 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 98

- "Editing a Named Configuration's Properties" on page 96
- "Configurations" on page 93

Working with Named Configurations

- "To Create a Named Configuration" on page 96
- "Editing a Named Configuration's Properties" on page 96
- "To Edit Port Numbers for Instances Referencing a Configuration" on page 98
- "To view a Named Configuration's Targets" on page 98
- "To Delete a Named Configuration" on page 99

▼ 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 93
- "The default-config Configuration" on page 94
- "Editing a Named Configuration's Properties" on page 96
- "To Edit Port Numbers for Instances Referencing a Configuration" on page 98
- "To view a Named Configuration's Targets" on page 98
- "To Delete a Named Configuration" on page 99

Editing a Named Configuration's Properties

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 93
- "To Create a Named Configuration" on page 96
- "To view a Named Configuration's Targets" on page 98
- "To Delete a Named Configuration" on page 99

▼ 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 95
- "Configurations" on page 93
- "Editing a Named Configuration's Properties" on page 96

▼ 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 95
- "Configurations" on page 93
- "To Create a Named Configuration" on page 96
- "Editing a Named Configuration's Properties" on page 96
- "To Delete a Named Configuration" on page 99

▼ 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 93
- "To Create a Named Configuration" on page 96
- "Editing a Named Configuration's Properties" on page 96
- "To view a Named Configuration's Targets" on page 98

CHAPTER 6

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 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 GlassFish Communications Server instances, if:

- Each server instance has access to the same session state data. Communications 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 56.

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.

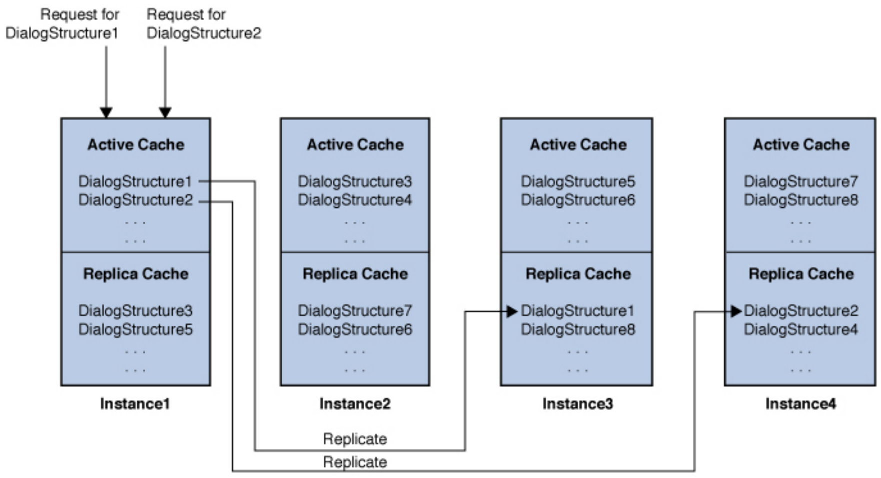
- Each server instance has the same distributable web application deployed to it. The `web-app` element of the `web.xml` deployment descriptor file or the `sip-app` element of the `sip.xml` deployment descriptor file must contain the `distributable` element.
- The web application uses high-availability 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)`.

How SIP Session Replication Works

The SIP Session Replication scheme enables Communications Server server to provide high availability and failover.

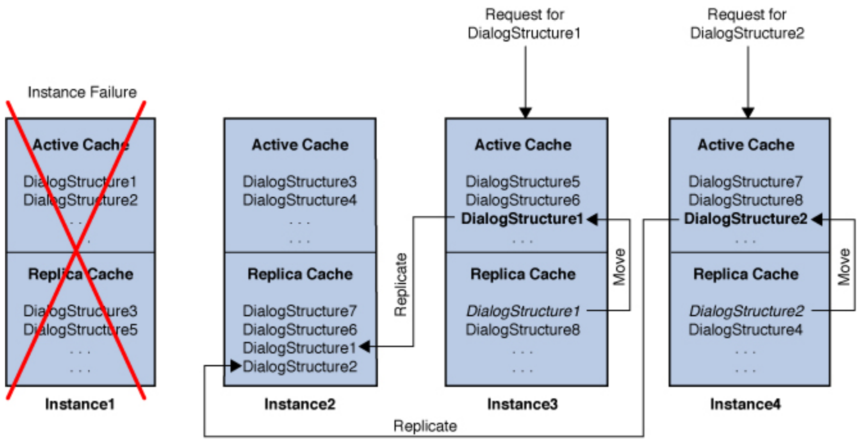
Consider a cluster with four instances. Each instance has an active cache of SIP Dialog Structures and a replica cache of SIP Dialog Structures. Each Dialog Structure is replicated based on its BEKey. Dialog Structure replicas are created based on the Converged Load Balancer's routing algorithm.

Note – A SIP Dialog Structure is a peer-to-peer SIP relationship between two user agents that persists for a specific time.



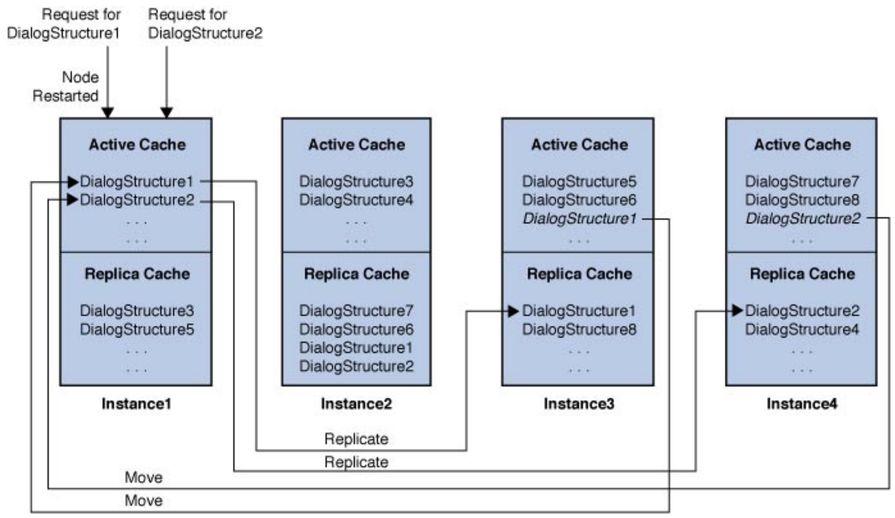
In this scheme, consider a scenario in which instance4 fails. The converged load balancer routes the request for DialogStructure1 to instance3, and the request for DialogStructure2 gets routed to instance4.

When the request for DialogStructure1 arrives in instance3, the replica cache of instance3 will already have DialogStructure1 in its replica cache. Therefore, DialogStructure1 is moved to the active cache of instance3 without requiring a network load for DialogStructure1. After DialogStructure1 is loaded to instance3's active cache, DialogStructure1 is replicated to its new partner as determined by the CLB algorithm (say instance2, as shown in the following figure.



The same replication approach is used for DialogStructure2 as well. instance2 is shown as the new replica partner for DialogStructure2 , but it could be instance3 or any other instance as computed by the converged load balancer algorithm.

When the failed node, instance1, gets restarted, the requests for DialogStructure1 and DialogStructure2 get remapped to instance1, as shown in the following figure:



When the instance1 gets the request for DialogStructure1, it moves the DialogStructure1 from instance3's active cache to its active cache. After DialogStructure1 is moved, instance1 replicates DialogStructure1to its replica partner as determined by the converged load balancer routing algorithm (which is instance3, as shown in the figure). Similarly, when the request for DialogStructure2 remaps to instance1, DialogStructure2 is moved from instance4 to instance1. DialogStructure2 is then replicate in instance4.

Note – After instance1 is restarted, the converged load balancer does not route the full traffic immediately. Instead, it routes the incremental traffic by using load-factor settings. After a specific interval (as determined by the load-factor setting), the restarted instance starts receiving full traffic.

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 105
- "Enabling Session Availability" on page 106

▼ 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 GlassFish Communications Server 2.0 Application Deployment Guide*.

1. **Create an Communications Server cluster.**

   For more information, see "To Create a Cluster" on page 60 .

2. **Set up load balancing for the cluster.**

   For more information , see 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 106.
   - Use the `asadmin` command-line utility. See `set(1)`.

4. **Restart each server instance in the cluster.**

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 116

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 116

   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 106 .
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 107
   - "Configuring Availability for the SIP Container" on page 110
   - "Configuring Availability for the EJB Container" on page 114
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 and SIP 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 108**

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 112

Configuring Availability for Individual Web Applications

To enable and configure availability for an individual web or converged web or 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 GlassFish Communications Server 2.0 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 110.**
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 112.

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 GlassFish Communications Server 2.0 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 GlassFish Communications Server 2.0 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.

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 107. 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 117

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 Server instance is stopped, the session data in the persistence store might not be cleared. To prevent this, undeploy the SFSB while the Communications 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 115**
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 GlassFish Communications Server 2.0 Application Deployment Guide*.

EXAMPLE 6–1 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 6–2 Example of EJB Deployment Descriptor Specifying Methods Checkpointing

```
<sun-ejb-jar>
    ...
    <enterprise-beans>
```

EXAMPLE 6–2 Example of EJB Deployment Descriptor Specifying Methods Checkpointing *(Continued)*

```
        ...
        <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 7

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 Server. It contains the following topics:

- "Overview of Java Message Service" on page 119
- "Configuring the Java Message Service" on page 120
- "Connection Pooling and Failover" on page 123
- "Using MQ Clusters with Communications Server" on page 124

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 Server, enabling you to create components such as message-driven beans (MDBs).

MQ is integrated with Communications 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 Server exchange JMS messages using the JMS provider integrated via the connector module. Creating a JMS resource in Communications 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 GlassFish Communications Server 2.0 Administration Guide*. For more information on JMS, see Chapter 18, "Using the Java Message Service," in *Sun GlassFish Communications Server 2.0 Developer's Guide*. For more information on connectors (resource adapters), see Chapter 12, "Developing Connectors," in *Sun GlassFish Communications Server 2.0 Developer's Guide*.

For more information about the Sun GlassFish 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 GlassFish Communications 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 GlassFish Communications Server 2.0 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 get`command 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 GlassFish Communications Server 2.0 Administration Guide*.

Note – You must restart the Communications 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 GlassFish Communications Server 2.0 Administration Guide*

Java Message Service Integration

MQ can be integrated with Communications 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 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 appliciation server process. Communications 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 Server instances and Message Queue brokers, set the type to LOCAL and give each Communications Server instance a different default JMS host. You can do this regardless of whether clusters are defined in the Communications 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 GlassFish Message Queue Administration Guide*.

In this case, Communications Server will use an externally configured broker or broker cluster. Also, you must start and stop MQ brokers separately from Communications Server, and use MQ tools to configure and tune the broker or broker cluster. REMOTE type is most suitable for Communications 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 Communications Server and the JMS broker are co-located in the same VM and the JMS service is started in-process and managed by the Communications 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 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 GlassFish 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 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 GlassFish 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 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 Server supports JMS connection pooling and failover. The Sun GlassFish Communications Server pools JMS connections automatically. When the Address List Behavior attribute is `random` (the default), Communications 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 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 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 GlassFish Communications Server 2.0 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 GlassFish Communications Server 2.0 Application Deployment Guide*.

The Communications 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 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 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 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 GlassFish Communications Server 2.0 Administration Guide*.

Using MQ Clusters with Communications 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 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 Server and MQ deployment, see the Planning Message Queue Broker Deployment chapter in the Sun GlassFish Enterprise Server 2.1 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, Communications 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 Communications 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 a cluster, if one does not already exist.**

For information on creating clusters, see "To Create a Cluster" on page 60.

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 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 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 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 instances.
- When all MQ brokers are down, it takes 30 minutes for Communications 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 8

RMI-IIOP Load Balancing and Failover

This chapter describes using high-availability features for remote EJB references and JNDI objects over RMI-IIOP.

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 Server supports dynamically reconfigured clusters. If the Communications Server instance on which the application client is deployed participates in a cluster, the Communications 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 GlassFish Communications 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 instances.
- Java EE applications are deployed to all instances and clusters that participate in load balancing.
- RMI-IIOP client applications are enabled for load balancing.

Communications 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 131.

Note – Communications Server does not support RMI-IIOP load balancing and failover over secure sockets layer (SSL).

Algorithm

Communications 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 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 Server instance serving the IIOP endpoint assigned to the `InitialContext` object. The references to those beans contain the IIOP endpoint addresses of all Communications 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 GlassFish Communications Server 2.0 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 GlassFish Communications Server 2.0 Administration Guide*.

Use the endpoints that you obtained in Step 6.

If the Communications 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 GlassFish Communications Server 2.0 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 8–1 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