Sun Java System Application Server Enterprise Edition 8.1 2005Q2 High Availability Administration Guide

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

Part No: 819–2555–12
August 2007

070830@18135

Contents

Figures

Tables

Examples

Preface

The *High Availability Administration Guide* describes the high-availability features of Sun Java™ SystemApplication Server, including how to:

- Install, configure, and administer High Availability Database (HADB).
- Install, configure, and use the HTTP Load Balancer Plug-in.
- Use named configurations to share server configuration attributes.
- Set up and use highly-available clusters.
- Configure node agents.
- Configure and use high-availability session persistence.
- Use other high availability features such as Java Message service and RMI-IIOP failover.

Who Should Use This Book

This guide is intended for system administrators in production environments. It assumes you are familiar with:

- Basic system administration
- Installing software
- Using a web browser
- Issuing commands in a terminal window

Before You Read This Book

Application Server can be purchased by itself or as a component of Sun Java™ Enterprise System (Java ES), a software infrastructure that supports enterprise applications distributed across a network or Internet environment. If you purchased Application Server as a component of Java ES, you should be familiar with the system documentation at `http://docs.sun.com/coll/1286.1`.

How This Book Is Organized

Chapter 1, "Application Server High Availability Features" provides an overview of Application Server's high availability features.

Chapter 2, "Installing and Setting Up High Availability Database" describes how to install and set up High Availability Database.

Chapter 3, "Administering High Availability Database" explains how to administer High Availability Database.

Chapter 4, "Configuring Load Balancing and Failover" describes how to install, configure, and use the HTTP Load Balancer Plug-in.

Chapter 5, "Using Application Server Clusters" explains Application Server clusters and how to configure and administer them.

Chapter 6, "Managing Named Configurations" explains how to use named configurations to share Application Server configuration attributes.

Chapter 7, "Configuring Node Agents" describes node agents and how to administer them.

Chapter 8, "Configuring High Availability Session Persistence and Failover" explains how to set up high-availability session persistence.

Chapter 9, "Java Message Service Load Balancing and Failover" describes Java Message Service Load Balancing and Failover.

Chapter 10, "RMI-IIOP Load Balancing and Failover" describes RMI-IIOP Load Balancing and Failover.

Application Server Documentation Set

The Application Server documentation set describes deployment planning and system installation. The stand-alone Application Server documentation is at `http://docs.sun.com/app/docs/coll/1310.1`. For an introduction to Application Server, refer to the books in the order in which they are listed in the following table.

TABLE P–1 Books in the Application Server Documentation Set

Book Title	Description
Release Notes	Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of the supported hardware, operating system, JDK, and JDBC/RDBMS.

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

Book Title	Description
Quick Start Guide	How to get started with the Application Server product.
Installation Guide	Installing the software and its components.
Deployment Planning Guide	Evaluating your system needs and enterprise to ensure that you deploy the Application Server in a manner that best suits your site. General issues and concerns that you must be aware of when deploying the server are also discussed.
Developer's Guide	Creating and implementing Java 2 Platform, Enterprise Edition (J2EE™ platform) applications intended to run on the Application Server that follow the open Java standards model for J2EE components and APIs. Includes general information about developer tools, security, assembly, deployment, debugging, and creating lifecycle modules.
J2EE 1.4 Tutorial	Using J2EE 1.4 platform technologies and APIs to develop J2EE applications.
Administration Guide	Configuring, managing, and deploying Application Server subsystems and components from the Administration Console.
High Availability Administration Guide	Post-installation configuration and administration instructions for the high-availability database.
Administration Reference	Editing the Application Server configuration file, `domain.xml`.
Upgrade and Migration Guide	Migrating your applications to the new Application Server programming model, specifically from Application Server 6.x and 7. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
Performance Tuning Guide	Tuning the Application Server to improve performance.
Troubleshooting Guide	Solving Application Server problems.
Error Message Reference	Solving Application Server error messages.
Reference Manual	Utility commands available with the Application Server; written in man page style. Includes the `asadmin` command line interface.

Related Books

For other Sun Java System server documentation, see:

- Message Queue documentation
- Directory Server documentation
- Web Server documentation

Documentation of Java ES and its components is at
`http://docs.sun.com/prod/entsys.05q4`.

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

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

domain-root-dir	Represents the directory containing all domains.	Sun Java Enterprise System installations on the Solaris platform:
		`/var/opt/SUNWappserver/domains/`
		Sun Java Enterprise System installations on the Linux platform:
		`/var/opt/sun/appserver/domains/`
		All other installations:
		install-dir`/domains/`

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

Typographic Conventions

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

TABLE P–3 Typographic Conventions

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

Symbol Conventions

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

TABLE P–4 Symbol Conventions

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

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.

Accessing Sun Resources Online

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

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

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

Third-Party Web Site References

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

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

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 819-2555.

◆ ◆ ◆ CHAPTER 1

Application Server High Availability Features

This chapter describes the high availability features in the Sun Java SystemApplication Server Enterprise Edition, with the following topics:

Overview of High Availability

High availability applications and services provide their functionality continuously, regardless of hardware and software failures. Application Server provides high availability for HTTP requests and session data (both HTTP session data and stateful session bean data).

Application Server provides high availability through the following sub-components and features:

Load Balancer Plug-in

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

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

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

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

For information on configuring load balancing and failover for see Chapter 4, "Configuring Load Balancing and Failover"

High Availability Database

Application Server Enterprise Edition provides the High Availability Database (HADB) for high availability storage of HTTP session and stateful session bean data. HADB is designed to support up to 99.999% service and data availability with load balancing, failover, and state recovery. Generally, you must configure and manage HADB independently of Application Server.

Keeping state management responsibilities separated from Application Server has significant benefits. Application Server instances spend their cycles performing as a scalable and high performance Java™ 2 Platform, Enterprise Edition (J2EE™ platform) containers delegating state replication to an external high availability state service. Due to this loosely coupled architecture, application server instances can be very easily added to or deleted from a cluster. The HADB state replication service can be independently scaled for optimum availability and performance. When an application server instance also performs replication, the performance of J2EE applications can suffer and can be subject to longer garbage collection pauses.

For information on planning and setting up your application server installation for high availability with HADB, including determining hardware configuration, sizing, and topology, see "Planning for Availability" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide* and Chapter 3, "Selecting a Topology," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide*.

Highly Available Clusters

A *cluster* is a collection of Application Server instances that work together as one logical entity. A cluster provides a runtime environment for one or more J2EE 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 J2EE application EAR file, a web module WAR file, or an EJB JAR file).
- Have the same set of resources, resulting in the same JNDI namespace.

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

Clusters and Configurations

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

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

Clusters, Instances, Sessions, and Load Balancing

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

- A server instance is not required to be part of a cluster. However, an instance that is not part of a cluster cannot take advantage of high availability through transfer of session state from one instance to other instances.
- The server instances within a cluster can be hosted on one or multiple machines. You can group server instances across different machines into a cluster.
- A particular load balancer can forward requests to server instances on multiple clusters. You can use this ability of the load balancer to perform an online upgrade without loss of service. For more information, see “Using Multiple Clusters for Online Upgrades Without Loss of Service” in the chapter “Configuring Clusters”

- A single cluster can receive requests from multiple load balancers. If a cluster is served by more than one load balancer, you must configure the cluster in exactly the same way on each load balancer.
- Each session is tied to a particular cluster. Therefore, although you can deploy an application on multiple clusters, session failover will occur only within a single cluster.

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

More Information

For information about planning a high-availability deployment, including assessing hardware requirements, planning network configuration, and selecting a topology, see *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide*. This manual also provides a high-level introduction to concepts such as:

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

For more information about developing and deploying applications that take advantage of high availability features, see *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Developer's Guide*.

Tuning High Availability Servers and Applications

For information on how to configure and tune applications and Application Server for best performance with high availability, see *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Performance Tuning Guide*, which discusses topics such as:

- Tuning persistence frequency and persistence scope
- Checkpointing stateful session beans
- Configuring the JDBC connection pool
- Session size
- Tuning HADB disk use, memory allocation, performance, and operating system configuration
- Configuring load balancer for best performance

High Availability Session Persistence

J2EE 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 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, the Application Server provides the capability to persist session state in the HADB. If the application server instance hosting the user session experiences a failure, the session state can be recovered, and the session can continue without loss of information.

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

◆ ◆ ◆ CHAPTER 2

Installing and Setting Up High Availability Database

This chapter covers the following topics:

Overview of High Availability Database

This section introduces the high availability database (HADB) and describes how to set up and configure HADB for use with the Application Server.

This section contains the following topics:

HADB and Application Server

HADB is a horizontally-scalable database that can be run and managed independently of the application server tier. It is designed to support up to 99.999% service and data availability with load balancing, failover, and state recovery capabilities.

Application Server uses HADB to store HTTP and stateful session bean (SFSB) session data. Without a session persistence mechanism, the HTTP or SFSB session state data is lost when a web or EJB container fails over.

Keeping state management separate from Application Server has significant benefits. Application Server instances spend their cycles performing as a scalable and high performance

Java™ 2 Platform, Enterprise Edition (J2EE™ platform) containers delegating state replication to an external high availability state service. Due to this loosely coupled architecture, you can easily add application server instances to and remove instances from a cluster. You can independently scale HADB state replication service for optimum availability and performance.

HADB Server Architecture

High availability means availability despite planned outages for upgrades or unplanned outages caused by hardware or software failures. HADB is based on a simple data model and redundant, scalable, and high performance technology. HADB offers an ideal platform for delivering all types of session state persistence within a high performance enterprise application server environment.

The following figure shows the architecture of a database with four active nodes and two spare nodes. Nodes 0 and 1 are a mirror node pair, as are nodes 2 and 3.

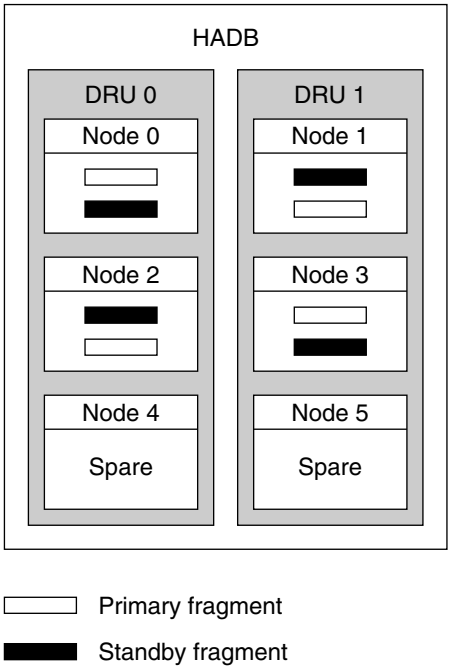


FIGURE 2–1 HADB Architecture

HADB achieves high data availability through fragmentation and replication of data. All tables in the database are partitioned to create subsets of approximately the same size called fragments. Fragmentation is based on a hash function that evenly distributes the data among the nodes of the database. Each fragment is stored twice in the database, in mirror nodes. This

ensures fault tolerance and fast recovery of data. In addition, if a node fails, or is shut down, a spare node can take over until the node is active again.

HADB nodes are organized into two Data Redundancy Units (DRUs), which mirror each other. Each DRU consists of half of the active and spare nodes, and contains one complete copy of the data. To ensure fault tolerance, the computers that support one DRU must be completely self-supported with respect to power (use of uninterruptible power supplies is recommended), processing units, and storage. If a power failure occurs in one DRU, the nodes in the other DRU can continue servicing requests until the power returns.

Without a session persistence mechanism, the HTTP or SFSB session state, including the passivated session state, is lost when one web or EJB container fails over to another. Use of the HADB for session persistence overcomes this situation. The HADB stores and retrieves state information in a separate but well-integrated persistent storage tier.

HADB reclaims space when session data is deleted. HADB places session data records in fixed size blocks. When all records of a block are deleted, the block is freed. Records of a block can be deleted randomly, creating *holes* in the block. When a new record is inserted into a block and contiguous space is needed, the holes are removed and thus the block is compacted.

This is a brief summary of the architecture. For more information, see *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide*.

HADB Nodes

A database node consists of a set of processes, a dedicated area of shared memory, and one or more secondary storage devices. The database stores, updates, and retrieves session data. Each node has a mirror node, therefore nodes occur in pairs. In addition, to maximize availability, include two or more spare nodes, one in each DRU, so if a node fails a spare can take over while the node is repaired.

For an explanation of node topology alternatives, see Chapter 3, "Selecting a Topology," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide*.

New Features and Improvements

The version of HADB provided with Sun Java System Application Server Enterprise Edition 8.1 has many new features and improvements.

HADB management is improved by changing the underlying components of the management system. The old `hadbm` interface functions are maintained with minor modifications. These changes also remove the dependency on SSH/RSH.

The management agent server process (`ma`) constitutes a domain and keeps the database configuration in a repository. The repository information is distributed among all agents.

The following topics provide more details:

- "General Improvements" on page 32
- "Specific Changes" on page 32

General Improvements

This version of HADB has the following general improvements:

- HADB no longer requires SSH/RSH.
- Administrator password for HADB management enhances security.
- Automatic online upgrade to future versions.
- Dependency on a single host is removed.
- Heterogeneous configurations of the database is supported. The device paths and history paths can be set individually.
- Ability to manage multiple platforms uniformly.

Specific Changes

This version of HADB includes the following changes from the previous version.

- UDP multicast is now required for network configuration.
- The management agent, `ma`, is now required to be running on all HADB hosts.
- New `hadbm` commands for domain management: `hadbm createdomain`, `hadbm deletedomain`, `hadbm extenddomain`, `hadbm reducedomain`, `hadbm listdomain`, `hadbm disablehost`. New commands for package management: `hadbm registerpackage`, `hadbm unregisterpackage`, `hadbm listpackage`
- All `hadbm` commands have the following new options:
  - `adminpassword`
  - `adminpasswordfile`
  - no-adminauthentication
  - `agent`
  - `javahome`

  Changes made to `hadbm create`:

  - New options:
    - `no-clear`
    - `no-cleanup`
    - `package`
    - `packagepath`
    - `agent`

    Extended options

- `hosts` (registers hosts in the domain).
- `set`

Options removed:

- `inetd`
- `inetdsetupdir`
- `configpath`
- `installpath`
- `set TotalDataDevideSizePerNode`
- `set managementProtocol`

Modified: `devicesize` is now optional, not required.

The `hadbm startnode` and `hadbm restartnode` commands' `startlevel` option has a new value, `clear` .

Changes made to `hadbm addnodes`: New options: `set`, `historypath`, `devicepath`. The `inetdsetupdir` option was removed.

Changes made to `hadbm get` and `hadbm set`: New attributes are `historypath` (heterogeneous path for history files) and `packagename`. Attributes eliminated are: `managementProtocol`, `TotalDeviceSizePerNode`, `installpath`, and `syslogging`.

Using Customer Support for HADB

Before calling customer support about HADB issues, gather as much of the following information as possible:

- System use profile:
  - Number of active concurrent users
  - Number of passive users
  - Number of users entering the system per second
  - Average session size
  - Session state timeout period (`SessionTimeout` value)
  - Transaction rate per user per second

  Machine properties:

  - RAM
  - Number of CPUs
  - CPU speed
  - Operating system version
  - Number of physical disks
  - Total disk size
  - Available disk space

- Data transfer capacity

Network properties:

- Transfer capacity
- Number of host names (network interfaces) per node

HADB data:

- History files
- `cfg` and `meta` files, located in `dbconfigpath`/*databasename*/*nodeno* directory. `dbconfigpath` is defined in the variable `ma.server.dbconfigpath` in the management agent configuration file.
- Version information (`hadbm --version`)

Preparing for HADB Setup

This section discusses the following topics:

- "Prerequisites" on page 34
- "Configuring Shared Memory and Semaphores" on page 38
- "Configuring Network Redundancy" on page 35
- "Synchronizing System Clocks" on page 40
- "File System Support" on page 40

After performing these tasks, see Chapter 3, "Administering High Availability Database."

For the latest information on HADB, see *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Release Notes*.

Prerequisites

Before setting up and configuring HADB, make sure your environment meets the following requirements:

- IPv4 is enabled. HADB supports IPv4 only. Disable IPv6 on the interfaces being used for HADB.
- The network (routers, switches, and network interfaces on the hosts) must be configured for User Datagram Protocol (UDP) multicast. If HADB hosts span multiple subnets, configure routers between the subnets to forward UDP multicast messages between the subnets.
- Configure any firewalls located between HADB hosts, or between HADB and Application Server hosts to allow all UDP traffic, both ordinary and multicast.

- Do not use dynamic IP addresses (assigned by Dynamic Host Configuration Protocol, or DHCP) for hosts that are part of `hadbm createdomain`, `hadbm extenddomain`, `hadbm create`, or `hadbm addnodes` commands.

Configuring Network Redundancy

Configuring a redundant network will enable HADB to remain available, even if there is a single network failure. You can configure a redundant network in two ways:

- On Solaris 9, you can set up network multipathing.
- Configure a double network, supported on all platforms except Windows Server 2003.

Setting Up Network Multipathing

Before setting up network multipathing, refer to the Administering Network Multipathing section in the *IP Network Multipathing Administration Guide*.

▼ To configure HADB host machines that already use IP multipathing:

1 Set network interface failure detection time.

For HADB to properly support multipathing failover, the network interface failure detection time must not exceed one second (1000 milliseconds), as specified by the `FAILURE_DETECTION_TIME` parameter in `/etc/default/mpathd`. Edit the file and change the value of this parameter to 1000 if the original value is higher:

```
FAILURE_DETECTION_TIME=1000
```

To put the change into effect, use this command:

```
pkill -HUP in.mpathd
```

2 Set up IP addresses to use with HADB.

As described in the *IP Network Multipathing Administration Guide*, multipathing involves grouping physical network interfaces into multipath interface groups. Each physical interface in such a group has two IP addresses associated with it:

- a physical interface address used for transmitting data.
- a test address for Solaris internal use only.

Specify only one physical interface address from the multipath group when you use `hadbm create --hosts`.

Example 2–1 Setting up Multipathing

Suppose there are two host machines named host1 and host2. If they each have two physical network interfaces, then set up the two interfaces as a multipath group. Run `ifconfig -a` on each host.

The output on host1 is:

```
bge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 5 inet 129.159.115.10 netmask ffffff00 broadcast 129.159.115.255
groupname mp0

bge0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 5 inet 129.159.115.11 netmask ffffff00 broadcast 129.159.115.255

bge1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 6 inet 129.159.115.12 netmask ffffff00 broadcast 129.159.115.255
groupname mp0

bge1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 6 inet 129.159.115.13 netmask ff000000 broadcast 129.159.115.255
```

The output on host2 is:

```
bge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 3 inet 129.159.115.20 netmask ffffff00 broadcast 129.159.115.255
groupname mp0

bge0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 3 inet 129.159.115.21 netmask ff000000 broadcast 129.159.115.255

bge1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 4 inet 129.159.115.22 netmask ffffff00 broadcast 129.159.115.255
groupname mp0

bge1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 4 inet 129.159.115.23 netmask ff000000 broadcast 129.159.115.255
```

In this example, the physical network interfaces on both hosts are listed after `bge0` and `bge1`. Those listed after `bge0:1` and `bge1:1` are multipath test interfaces (marked DEPRECATED in the `ifconfig` output), as described in the *IP Network Multipathing Administration Guide*.

To set up HADB in this environment, select one physical interface address from each host. In this example, HADB uses IP address 129.159.115.10 from host1 and 129.159.115.20 from host2. To create a database with one database node per host, use the command `hadbm create --host`. For example

```
hadbm create --host 129.159.115.10,129.159.115.20
```

To create a database with two database nodes on each host, use the command:

```
hadbm create --host 129.159.115.10,129.159.115.20,
129.159.115.10,129.159.115.20
```

In both cases, you must configure the agents on host1 and host2 with separate parameters to specify which interface on the machines the agents should use. So, on host1 use:

```
ma.server.mainternal.interfaces=129.159.115.10
```

And on host2 use:

```
ma.server.mainternal.interfaces=129.159.115.20
```

For information on the `ma.server.mainternal.interfaces` variable, see "Configuration File" on page 53.

Configuring Double Networks

To enable HADB to tolerate single network failures, use IP multipathing if the operating system (for example, Solaris) supports it. Do not configure HADB with double networks on Windows Server 2003—the operating system does not work properly with double networks.

If your operating system is not configured for IP multipathing, and HADB hosts are equipped with two NICs, you can configure HADB to use double networks. For every host, the IP addresses of each of the network interface card (NIC) must be on separate IP subnets.

Within a database, all nodes must be connected to a single network, or all nodes must be connected to two networks.

Note – Routers between the subnets must be configured to forward UDP multicast messages between subnets.

When creating an HADB database, use the –hosts option to specify two IP addresses or host names for each node: one for each NIC IP address. For each node, the first IP address is on net-0 and the second on net-1. The syntax is as follows, with host names for the same node separated by a plus sign (+):

```
--hosts=node0net0name+node0net1name
,node1net0name+node1net1name
,node2net0name+node2net1name
, ...
```

For example, the following argument creates two nodes, each with two network interfaces. The following host option is used to create these nodes:

```
--hosts 10.10.116.61+10.10.124.61,10.10.116.62+10.10.124.62
```

Thus, the network addresses

- For node0 are `10.10.116.61` and `10.10.124.61`
- For node1 are `10.10.116.62` and `10.10.124.62`

Notice that `10.10.116.61` and `10.10.116.62` are on the same subnet, and `10.10.124.61` and `10.10.124.62` are on the same subnet.

In this example, the management agents must use the same subnet. Thus, the configuration variable `ma.server.mainternal.interfaces` must be set to, for example, `10.10.116.0/24`. This setting can be used on both agents in this example.

Configuring Shared Memory and Semaphores

You must configure shared memory and semaphores before installing HADB. The procedure depends on your operating system.

▼ To configure shared memory and semaphores on Solaris

1 Log in as root.

2 Configure shared memeory.

Set the value of `shmmax` to the size of the physical memory on the HADB host machine. The maximum shared memory segment size must be larger than the size of the HADB database buffer pool. For example, for a machine with a 2 GByte (0x8000000 hexadecimal) main memory, add the following to the `/etc/system` file:

```
set shmsys:shminfo_shmmax=0x80000000
 set shmsys:shminfo_shmseg=20
```

On Solaris 9 and later, `shmsys:shminfo_shmseg` is obsolete.

Set `shminfo_shmmax` to the total memory in your system (in hexadecimal notation the value `0x80000000` shown is for 2 Gigabytes of memory).

Note – Specify the value of `shmsys:shminfo_shmmax` using the hexadecimal value for the memory size. To determine your host's memory, use this command:

```
prtconf | grep Memory
```

---

3 **Configure semaphores.**

Check the `/etc/system` file for semaphore configuration entries. This file might already contain `semmni`, `semmns`, and `semmnu` entries. For example:

```
set semsys:seminfo_semmni=10
set semsys:seminfo_semmns=60
set semsys:seminfo_semmnu=30
```

If the entries are present, increment the values by adding 16, 128, and 1000 respectively. So, the entries in the example above would change to:

```
set semsys:seminfo_semmni=26
set semsys:seminfo_semmns=188
set semsys:seminfo_semmnu=1030
```

If the `/etc/system` file does not these entries, add them at the end of the file:

```
set semsys:seminfo_semmni=16
set semsys:seminfo_semmns=128
set semsys:seminfo_semmnu=1000
```

This is sufficient to run up to 16 HADB nodes on the computer. For information on setup for more than 16 nodes, see the HADB chapter in the *Sun Java System Application Server Enterprise Edition 8.1 2005Q1 Performance Tuning Guide*.

4 **Reboot the machine.**

▼ To configure shared memory on Linux

1 **Log in as root.**

2 **Edit the file** `/etc/sysctl.conf`

3 **Set the** `kernel.shmax` **and** `kernel.shmall` **parameters.**

The `kernel.shmax` parameter defines the maximum size in bytes for a shared memory segment. The `kernel.shmall` parameter sets the total amount of shared memory in pages that can be used at one time on the system. Set the value of both of these parameters to the amount physical memory on the machine. Specify the value as a decimal number of bytes. For example, for a machine having 512 Mbytes of physical memory:

```
kernel.shmmax=536870912
 kernel.shmall=536870912
```

4 **Reboot the machine. using this command:**

```
sync; sync; reboot
```

Procedure for Windows

Windows does not require any special system settings. However, if you want to use an existing J2SE installation, set the `JAVA_HOME` environment variable to the location where the J2SE is installed.

Synchronizing System Clocks

You must synchronize clocks on HADB hosts, because HADB uses time stamps based on the system clock. HADB uses the system clock to manage timeouts and to time stamp events logged to history files. For troubleshooting, you must analyze all the history files together, since HADB is a distributed system. So, it is important that all the hosts' clocks be synchronized

Do not adjust system clocks on a running HADB system. Doing so can cause problems in the operating system or other software components that can in turn cause problems such as hangs or restarts of HADB nodes. Adjusting the clock backward can cause some HADB server processes to hang as the clock is adjusted.

To synchronize clocks:

- On Solaris, use `xntpd` (network time protocol daemon).
- On Linux, use `ntpd`.
- On Windows, use `NTPTime` on Windows

If HADB detects a clock adjustment of more than one second, it logs it to the node history file, for example:

```
NSUP INF 2003-08-26 17:46:47.975 Clock adjusted.
 Leap is +195.075046 seconds.
```

File System Support

This section describes some restrictions of HADB with certain file systems.

Red Hat Enterprise Linux

HADB supports the ext2 and ext3 file systems on Red Hat Enterprise Linux 3.0. For Red Hat Enterprise Linux 2.1, HADB supports the ext2 file system.

Veritas File System

When using the Veritas File System on Solaris, HADB writes the message `WRN: Direct disk I/O mapping failed` to the history files. This message indicates that HADB cannot turn on direct input/output (I/O) for the data and log devices. Direct I/O reduces the CPU cost of writing disk pages. It also reduces overhead of administering "dirty" data pages in the operating system.

To use direct I/O with Veritas File System, do one of the following:

- Create the data and log devices on a file system that is mounted with the option `mincache=direct`. This option applies to all files created on the file system. Check the `mount_vxfs(1M)` command for details.
- Use the Veritas Quick I/O facility to perform raw I/O to file system files. For details, see the *VERITAS File System 4.0 Administrator's Guide for Solaris*.

Note – These configurations have not been tested with the Sun Java System Application Server.

Installation

In general, you can install HADB on the same system as Application Server (co-located topology) or on separate hosts (separate tier topology). For more information on these two options, see Chapter 3, "Selecting a Topology," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide*. However, you must install the HADB management client to be able to set up high availability with the `asadmin ha-config-cluster` command. When using the Java Enterprise System installer, you must install an entire HADB instance to install the management client, even if the nodes are to be installed on a separate tier.

HADB Installation

On a single or dual CPU system, you can install both HADB and Application Server if the system has at least two Gbytes of memory. If not, install HADB on a separate system or use additional hardware. To use the `asadmin ha-configure-cluster command` , you must install both HADB and Application Server.

Each HADB node requires 512 Mbytes of memory, so a machine needs one Gbyte of memory to run two HADB nodes. If the machine has less memory, set up each node on a different machine. For example, you can install two nodes on:

- Two single-CPU systems, each with 512 Mbytes to one Gbyte of memory
- A single or dual CPU system with one Gbyte to two Gbytes of memory

  You can install HADB with either the Java Enterprise System installer or the Application Server standalone installer. In either installer, choose the option to install HADB (called High Availability Session Store in Java ES) in the Component Selection page. Complete the installation on your hosts. If you are using the Application Server standalone installer, and choose two separate machines to run HADB, you must choose an identical installation directory on both machines.

Default Installation Directories

Throughout this manual, *HADB_install_dir* represents the directory in which HADB is installed. The default installation directory depends on whether you install HADB as part of the Java Enterprise System. For Java Enterprise System, the default installation directory is `/opt/SUNWhadb/4`. For the standalone Application Server installer, it is `/opt/SUNWappserver/hadb/4`.

Node Supervisor Processes Privileges

The node supervisor processes (NSUP) ensure the availability of HADB by exchanging "I'm alive" messages with each other. The NSUP executable files must have root privileges so they can respond as quickly as possible. The `clu_nsup_srv` process does not consume significant CPU resources, has a small footprint, and so running it with real-time priority does not affect performance.

Note – The Java Enterprise System installer automatically sets the NSUP privileges properly, so you do not need to take any further action. However, with the standalone Application Server (non-root) installer, you must set the privileges manually before creating a database.

Symptoms of Insufficient Privileges

If NSUP executables do not have the proper privileges, you might notice symptoms of resource starvation such as:

- Performance issues or `HIGH LOAD` messages in the HADB history log.
- False network partitioning and node restarts, preceded by a warning "Process blocked for *x* seconds" in HADB history files.
- Aborted transactions and other exceptions.

Restrictions

If NSUP cannot set the real-time priority `errno` is set to EPERM on Solaris and Linux. On Windows it issues the warning "Could not set realtime priority". The error is written to the `ma.log` file, and the process continues without real-time priority.

Setting real-time priorities is not possible when:

- HADB is installed in Solaris 10 non-global zones
- `PRIV_PROC_LOCK_MEMORY` (allow a process to lock pages in physical memory) and/or `PRIV_PROC_PRIOCNTL` privileges are revoked in Solaris 10
- Users turn off setuid permission
- Users install the software as tar files (the non-root installation option for the Application Server)

▼ To Give Node Supervisor Processes Root Privileges

1. **Log in as root.**
2. **Change your working directory to** *HADB_install_dir*`/lib/server`**.**

   The NSUP executable file is `clu_nsup_srv` .
3. **Set the file's** `suid` **bit with this command:**

   ```
   chown root clu_nsup_srv
   ```
4. **Set the file's ownership to root with this command:**

   ```
   chmod u+s clu_nsup_srv
   ```

   This starts the `clu_nsup_srv` process as root, and enables the process to give itself realtime priority.

   To avoid any security impact, the real-time priority is set immediately after the process is started and the process falls back to the effective UID once the priority has been changed. Other HADB processes run with normal priority.

Setting up High Availability

This section provides the steps for creating a highly available cluster, and testing HTTP session persistence.

This section discusses the following topics:

Prerequisites

Before configuring HADB, do the following:

▼ To prepare your system for High Availability

1. **Install Application Server instances and the Load Balancer Plug-in.**

   For more information, see the *Java Enterprise System Installation Guide* (if you are using Java ES) or *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Installation Guide* (if you are using the standalone Application Server installer).

2 **Create Application Server domains and clusters.**

For more information, see the *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*.

3 **Install and configure your web server software.**

For more information, see "Configuring Web Servers for Load Balancing" on page 107

4 **Setup and configure load balancing.**

For more information, see "Setting Up HTTP Load Balancing" on page 105.

Starting the HADB Management Agent

The management agent, `ma`, executes management commands on HADB hosts and ensures availability of the HADB node supervisor processes by restarting them if they fail.

For a production deployment, start the management agent as a service to ensure its availability. This section provides abbreviated instructions for starting the management agent as a service with its default configuration.

For more details, including instructions on starting the management agent in console mode for testing or evaluation and information on customizing its configuration, see "Using the HADB Management Agent" on page 51.

This section describes how to start the management agent as a service with default configuration when using Java Enterprise System.

▼ To Start the Management Agent with Java Enterprise System on Solaris or Linux

1 **Create the following softlinks to the file `/etc/init.d/ma-initd`:**

```
/etc/rc0.d/K20ma-initd
/etc/rc1.d/K20ma-initd
/etc/rc2.d/K20ma-initd
/etc/rc3.d/S99ma-initd
/etc/rc5.d/S99ma-initd
/etc/rcS.d/K20ma-initd
```

2 **Reboot the machine.**

To deactivate automatic start and stop of the agent, remove the links or change the letters K and S in the link names to lowercase.

To Start the Management Agent with Java Enterprise System on Windows

1. **Open a command window.**
2. **Enter the command:** *HADB_install_dir*`\bin\ma -i`**.**

   This installs and starts the management agent with its default configuration.

Next Steps To stop the management agent and remove (deregister) it as a service, use the command: *HADB_install_dir*`\bin\ma -r`

To Start the Management Agent with Standalone Application Server on Solaris or Linux

1. **In a shell, change your current directory to** *HADB_install_dir* `/bin`**.**
2. **Edit the shell script** `ma-initd`**.**

   Replace the default values of HADB_ROOT and HADB_MA_CFG in the script to reflect your installation:

   - HADB_ROOT is the HADB installation directory, *HADB_install_dir*.
   - HADB_MA_CFG is the location of the management agent configuration file. For more information, see "Customizing Management Agent Configuration" on page 53
3. **Copy** `ma-initd` **to the directory** `/etc/init.d`
4. **Create the following soft links to the file** `/etc/init.d/ma-initd`**:**

```
/etc/rc0.d/K20ma-initd
/etc/rc1.d/K20ma-initd
/etc/rc2.d/K20ma-initd
/etc/rc3.d/S99ma-initd
/etc/rc5.d/S99ma-initd
/etc/rcS.d/K20ma-initd
```

To Start the Management Agent with Standalone Application Server on Windows

1. **Open a command window.**
2. **Enter the command:** *HADB_install_dir*`\bin\ma -i ma.cfg`

   Now if the process fails or the machine reboots, the management agent will automatically restart.

Next Steps To stop the management agent and remove (deregister) it as a service, use the command: *HADB_install_dir*`\bin\ma -r ma.cfg`

Configuring a Cluster for High Availability

Before starting this section, you must have created one or more Application Server clusters. For information on how to create a cluster, see "To Create a Cluster" on page 136.

From the machine on which the Domain Administration Server is running, configure the cluster to use HADB using this command:

`asadmin configure-ha-cluster --user admin --hosts` *hadb_hostname*`,`*hadb_hostname* **`--devicesize 256`** *clusterName*

Replace *hadb_hostname* with the host name of the machine where HADB is running, and *clusterName* with the name of the cluster. If you are using just one machine, you must provide the host name twice.

This simplified example runs two nodes of HADB on the same machine. In production settings, using more than one machine is recommended.

Configuring an Application for High Availability

In Admin Console, select the application under Applications > Enterprise Applications. Set Availability Enabled and then click Save.

Restarting the Cluster

To restart a cluster in Admin Console, choose Clusters > *cluster-name*. Click Stop Instances. Once the instances have stopped, click "Start Instances."

Alternatively, use these `asadmin` commands:

```
asadmin stop-cluster --user admin cluster-name
asadmin start-cluster --user admin cluster-name
```

For more information on these commands, see `stop-cluster`(1) and `start-cluster`(1).

Restarting the Web Server

To restart the Web Server, type this Web Server command:

```
web_server_root/https-hostname/reconfig
```

Replace *web_server_root* with your Web Server root directory and *hostname* with the name of your host machine.

▼ To Clean Up the Web Server Instance Acting as Load Balancer

1 **Delete the Load Balancer configuration:**

```
asadmin delete-http-lb-ref --user admin --config MyLbConfig FirstCluster

asadmin delete-http-lb-config --user admin MyLbConfig
```

2 **If you created a new Web Server instance you can delete it by:**

   a. **Log on to the Web Server's Administration Console.**

   b. **Stop the instance.**

      Delete the instance.

Upgrading HADB

HADB is designed to provide "always on" service that is uninterrupted by upgrading the software. This section describes how to upgrade to a new version of HADB without taking the database offline or incurring any loss of avaiability.

The following sections describe how to upgrade your HADB installation:

▼ To upgrade HADB to a newer version

1 **Install new version of HADB.**

2 **Unregister your existing HADB installationas described in "Unregistering HADB Packages" on page 49**

3 **Register the new HADB version, as described in "Registering HADB Packages" on page 48**

Registering the HADB package in the HADB management domain makes it easy to upgrade or change HADB packages. The management agent keeps track of where the software packages are located, as well as the version information for the hosts in the domain. The default package name is a string starting with V and containing the version number of the `hadbm` program.

4 **Change the package the database uses.**

Enter the following command:

```
hadbm set PackageName=package
```

where package is the version number of the new HADB pacakge.

5 **If neccessary, replace the management agent startup script.**

For more information, see "Replacing the Management Agent Startup Script" on page 50

Registering HADB Packages

Use the `hadbm registerpackage` command to register the HADB packages that are installed on the hosts in the management domain. HADB packages can also be registered when creating a database with `hadbm create`.

Before using the `hadm registerpackage` command, ensure that all management agents are configured and running on all the hosts in the hostlist, the management agent's repository is available for updates, and no software package is already registered with the same package name.

The command syntax is:

`hadbm registerpackage` `--packagepath=`*`path`* **`[`**`--hosts=`*`hostlist`***`]`**
`[``--adminpassword=`*`password`* **`|`** `--adminpasswordfile=`*`file`***`]`** **`[`**`--agent=`*`maurl`***`]`**
`[[`*`package-name`*`]]`

The *package-name* operand is the name of the package.

The following table describes the special `hadbm registerpackage` command option. See "Security Options" on page 60 and "General Options" on page 61 for a description of other command options.

TABLE 2–1 `hadbm registerpackage` Options

Option	Description
--hosts=*hostlist* -H	List of hosts, either comma-separated or enclosed in double quotes and space separated.
--packagepath=*path* -L	Path to the HADB software package.

For example, the following command registers software package v4 on hosts host1, host2, and host3:

```
hadbm registerpackage
--packagepath=hadb_install_dir/SUNWHadb/4.4
--hosts=host1,host2,host3 v4
```

The response is:

```
Package successfully registered.
```

If you omit the `--hosts` option, the command registers the package on all enabled hosts in the domain.

Unregistering HADB Packages

Use the `hadbm unregisterpackage` command to remove HADB packages that are registered with the management domain.

Before using the `hadbm unregisterpackage` command, ensure that all management agents are configured and running on all the hosts in the hostlist, the management agent's repository is available for updates, the package is registered in the management domain, and no existing databases are configured to run on the package about to be unregistered.

The command syntax is:

```
hadbm unregisterpackage
--hosts=hostlist
[--adminpassword=password | --adminpasswordfile= file]
[--agent= maurl]
[package-name ]
```

The *package-name* operand is the name of the package.

See "Registering HADB Packages" on page 48 above for a description of the `--hosts` option. If you omit the `--hosts` option, the hostlist defaults to the enabled hosts where the package is registered. See "Security Options" on page 60 and "General Options" on page 61 for a description of other command options.

EXAMPLE 2–2 Example of unregistering HADB

To unregister software package v4 from specific hosts in the domain:

```
hadbm unregisterpackage --hosts=host1,host2,host3 v4
```

The response is:

```
Package successfully unregistered.
```

Replacing the Management Agent Startup Script

When you install a new version of HADB, you may need to replace the management agent startup script in `/etc/init.d/ma-initd`. Check the contents of the file, *HADB_install_dir*`/lib/ma-initd`. If it is different from the old `ma-initd` file, replace the old file with the new file.

◆ ◆ ◆ CHAPTER 3

Administering High Availability Database

This chapter describes the high availability database (HADB) in the Sun Java System Application Server Enterprise Edition environment. It explains how to configure and administer the HADB. Before you can create and administer the HADB, you must first determine the topology of your systems and install the HADB software on the various machines.

This chapter discusses the following topics:

Using the HADB Management Agent

The management agent, `ma`, executes management commands on HADB hosts. The management agent also ensures availability of the HADB node supervisor processes by restarting them if they fail.

Management Agent Command Syntax

The syntax of the management agent `ma` command is:

```
ma [common-options]
[ service-options]
config-file
```

Where:

- *common-options* is one or more of the common options described in "Management Agent Command Syntax" on page 51.
- *service-options* is one of the Windows service options described in "Management Agent Command Syntax" on page 51.
- *config-file* is the full path to the management agent configuration file. For more information, see "Customizing Management Agent Configuration" on page 53.

TABLE 3–1 Management Agent Common Options

Option	Description	Default
--define *name*=*value*-D	Assign *value* to property *name*, where property is one of the properties defined in "Configuration File" on page 53. This option can be repeated multiple times.	None
--help-?	Display help information.	False
--javahome *path*-j	Use Java Runtime Environment (1.4 or later) located at *path*.	None
--systemroot *path*-y	Path to the operating system root as normally set in %SystemRoot%.	None
--version-V	Display version information.	False

"Management Agent Command Syntax" on page 51 describes options for starting the management agent as a Windows service. The -i, -r, and -s options are mutually exclusive; that is, use only one of them at a time.

On Windows, when specifying paths for property values in the configuration file or on the command line, escape file paths containing spaces with double quotes (`"`). Escape colon (`:`) drive separators, , and backslash (`\`) directory separators, with double quotes and a backslash, like this: `"\:` and `"\\`.

TABLE 3–2 Management Agent Service Options (Windows Only)

Option	Description	Default
--install-i	Install the agent as a Windows service and start the service. Use only one of -i, -r, and -s options.	False
--name *servicename*-n	Use specified name for the service when running multiple agents on a host.	HADBMgmtAgent

TABLE 3–2 Management Agent Service Options (Windows Only) *(Continued)*

Option	Description	Default
--remove-r	Stop the service and delete the agent from the Windows service manger. Use only one of -i, -r, and -s options.	False
--service-s	Run the agent as a Windows service. Use only one of -i, -r, and -s options.	False

Customizing Management Agent Configuration

HADB includes a configuration file that you can use to customize the management agent settings. When you start the management agent without specifying a configuration file, it uses default values. If you specify a configuration file, the management agent will use the settings in that file. You can re-use the configuration file on all hosts in a domain.

▼ To customize the management agent configuration on each HADB host

1. **Edit the management agent configuration file and set the values as desired.**
2. **Start the management agent, specifying the customized configuration file as the argument.**

Configuration File

With Java Enterprise System, all the entries in the configuration file are commented out. No changes are required to use the default configuration. To customize the management agent configuration, remove the comments from the file, change the values as desired, then start the management agent specifying the configuration file as an argument.

The management agent configuration file is installed to:

- Solaris and Linux: `/etc/opt/SUNWhadb/mgt.cfg`.
- Windows: *install_dir* `\lib\mgt.cfg`.

With the standalone installer, the management agent configuration file is installed to:

- Solaris and Linux: *HADB_install_dir* `/bin/ma.cfg` .
- Windows: *HADB_install_dir* `\bin\ma.cfg`.

The following table describes the settings in the configuration file.

TABLE 3–3 Configuration File Settings

Setting Name	Description	Default
console.loglevel	Console log level. Valid values are SEVERE, ERROR, WARNING, INFO, FINE, FINER, FINEST	WARNING
logfile.loglevel	Log file log level. Valid values are SEVERE, ERROR, WARNING, INFO, FINE, FINER, FINEST	INFO
logfile.name	Name and location of log file. Must be a valid path with read/write access.	Solaris and Linux:`/var/opt/SUNWhadb/ma/ma.log` Windows: *HADB_install_dir*`\ma.log`
ma.server.type	Client protocol. Only JMXMP is supported.	jmxmp
ma.server. jmxmp.port	Port number for internal (UDP) and external (TCP) communication. Must be a positive integer. Recommended range is 1024-49151.	1862
ma.server. mainternal.interfaces	Interfaces for internal communication for machines with multiple interfaces. Must be a valid IPv4 address mask. All management agents in a domain must use the same subnet For example, if a host has two interfaces, 10.10.116.61 and 10.10.124.61, use 10.10.116.0/24 to use the first interface. The number after the slash indicates the number of bits in the subnet mask.	None
ma.server. dbdevicepath	Path to store HADB device information.	Solaris and Linux: `/var/opt/SUNWhadb/4` Windows: *HADB_install_dir* `\device`
ma.server. dbhistorypath	Path to store HADB history files.	Solaris and Linux: `/var/opt/SUNWhadb` Windows: REPLACEDIR (replaced by the actual URL at runtime.)
ma.server. dbconfigpath	Path to store node configuration data.	Solaris and Linux: `/var/opt/SUNWhadb/dbdef` Windows: `C:\Sun\SUNWhadb\dbdef`
repository.dr.path	Path to domain repository files.	Solaris and Linux: `/var/opt/SUNWhadb/repository` Windows: `C:\Sun\SUNWhadb\repository`

Starting the Management Agent

You can start the management agent two ways:

- As a service, for production use. See "Starting the Management Agent as a Service" on page 55 To ensure availability of the management agent, make sure it is restarted automatically when the system reboots. See "Ensuring Automatic Restart of the Management Agent" on page 56.
- As a regular process (in console mode), for evaluation, testing, or development. See "Starting the Management Agent in Console Mode" on page 58.

  In each case, the procedures are different depending on whether you are using Java Enterprise System or the standalone Application Server.

Starting the Management Agent as a Service

Starting the management agent as a service ensures that it will continue to run until the system shuts down or you explicitly stop it.

Starting the Management Agent as a Service with Java Enterprise System on Solaris or Linux

To start the management agent as a service, use this command:

```
/etc/init.d/ma-initd start
```

To stop the service, use this command:

```
/etc/init.d/ma-initd stop
```

Starting the Management Agent as a Service with Java Enterprise System on Windows

To start the management agent as a Windows service, use this command:
HADB_install_dir`\bin\ma -i [`*config-file* `]`

The optional argument *config-file* specifies the management agent configuration file. Use a configuration file only if you want to change the default management agent configuration.

To stop the management agent and remove (deregister) it as a service, use the command:
HADB_install_dir`\bin\ma -r [`*config-file* `]`

To perform administration, choose Administrative Tools | Services, which enables you to start and stop the service, disable automatic startup, and so on.

Starting the Management Agent as a Service with Standalone Application Server on Solaris or Linux

To start the management agent as a service, use this command:

```
HADB_install_dir/bin/ma-initd start
```

To stop the service, use this command:

```
HADB_install_dir/bin/ma-initd stop
```

Starting the Management Agent as a Service with Standalone Application Server on Windows

To start the management agent as a Windows service, use this command:
HADB_install_dir`\bin\ma -i [`*config-file* `]`

The optional argument *config-file* specifies the management agent configuration file. Use a configuration file only if you want to change the default management agent configuration.

To stop the management agent and remove (deregister) it as a service, use the command:
HADB_install_dir`\bin\ma -r [`*config-file* `]`

To perform administration, choose Administrative Tools | Services, which enables you to start and stop the service, disable automatic startup, and so on.

Ensuring Automatic Restart of the Management Agent

On Windows platforms, once you have started the management agent as a service, use the Windows administrative tools to set the service Startup type to "Automatic," and the desired Recovery options.

On Solaris and Linux platforms, use the procedures in this section to ensure the availability of the management agent in case the `ma` process fails or the operating system reboots. Doing so is appropriate for a production deployment.

The following procedures ensure the management agent starts only when the system enters:

- Runlevel 3 on Solaris (the default).
- Runlevel 5 on RedHat Linux (the default in graphical mode).

Entering other runlevels stops the management agent.

To Configure automatic restart with Java Enterprise System on Solaris or Linux

Before You Begin This section assumes you have a basic understanding of operating system initialization and runlevels. For information on these topics, see your operating system documentation.

1. **Ensure that your system has a default runlevel of 3 or 5.**

   To check the default runlevel of your system, inspect the file `/etc/inittab` , and look for a line near the top similar to this:

   ```
   id:5:initdefault:
   ```

   This example shows a default runlevel of 5.

2. **Create the following softlinks to the file `/etc/init.d/ma-initd`:**

   ```
   /etc/rc0.d/K20ma-initd
   /etc/rc1.d/K20ma-initd
   /etc/rc2.d/K20ma-initd
   /etc/rc3.d/S99ma-initd
   /etc/rc5.d/S99ma-initd
   /etc/rcS.d/K20ma-initd
   ```

3. **Reboot the machine.**

Next Steps To deactivate automatic start and stop of the agent, remove the links or change the letters K and S in the link names to lowercase.

To Configure automatic restart with Standalone Application Server on Solaris or Linux

1. **In a shell, change your current directory to *HADB_install_dir* `/bin`.**

2. **Edit the shell script `ma-initd`.**

   Make sure the default values of HADB_ROOT and HADB_MA_CFG in the script to reflect your installation:

   - HADB_ROOT is the HADB installation directory, *HADB_install_dir.*
   - HADB_MA_CFG is the location of the management agent configuration file. For more information, see "Customizing Management Agent Configuration" on page 53

3. **Copy `ma-initd` to the directory `/etc/init.d`**

4 **Create the following soft links to the file** `/etc/init.d/ma-initd`**:**

```
/etc/rc0.d/K20ma-initd
/etc/rc1.d/K20ma-initd
/etc/rc2.d/K20ma-initd
/etc/rc3.d/S99ma-initd
/etc/rc5.d/S99ma-initd
/etc/rcS.d/K20ma-initd
```

Next Steps To deactivate automatic start and stop of the agent, remove the links or change the letters K and S in the link names to lowercase.

Starting the Management Agent in Console Mode

You may wish to start the management agent manually in console mode for evaluation or testing. Do not start the management agent this way in a production environment, because the `ma` process will not restart after a system or process failure and will terminate when the command window is closed.

Starting management agent in Console Mode with Java Enterprise System on Solaris or Linux

To start the HADB management agent in console mode, use the command:

```
opt/SUNWhadb/bin/ma [config-file]
```

The default management agent configuration file is `/etc/opt/SUNWhadb/mgt.cfg`

To stop the management agent, kill the process or close the shell window.

Starting management agent in Console Mode with Java Enterprise System on Windows

To start the management agent in console mode, use the command:

```
HADB_install_dir\bin\ma [config-file]
```

The optional argument *config-file* is the name of the management agent configuration file. For more information on the configuration file, see "Customizing Management Agent Configuration" on page 53.

To stop the agent, kill the process.

Starting management agent in Console mode with Standalone Application Server on Windows

To start the management agent in console mode, use the command:

```
HADB_install_dir\bin\ma [config-file]
```

The optional argument *config-file* is the name of the management agent configuration file; for more information, see "Customizing Management Agent Configuration" on page 53

To stop the management agent, kill the process.

Starting management agent in Console Mode with Standalone Application Server on Solaris or Linux

To start the HADB management agent in console mode, use the command:

```
HADB_install_dir/bin/ma [config-file]
```

The default management agent configuration file is *HADB_install_dir* `/bin/ma.cfg`

To stop the management agent, kill the process or close the shell window.

Using the hadbm Management Command

Use the `hadbm` command-line utility to manage an HADB domain, its database instances, and nodes. The `hadbm` utility (also called the management client) sends management requests to the specified management agent, acting as a management server, which has access to the database configuration from the repository.

This section describes the hadbm command-line utility, with the following topics:

Command Syntax

The `hadbm` utility is located in the *HADB_install_dir* `/bin` directory. The general syntax of the `hadbm` command is:

`hadbm` *subcommand*
[*-short-option* [*option-value*]]
[*--long-option* [*option-value*]]
[*operands*]

The subcommand identifies the operation or task to perform. Subcommands are case-sensitive. Most commands have one operand (usually `dbname`), but some have none, and some have two.

Options modify how `hadbm` performs a subcommand. Options are case-sensitive. Each option has a long form and a short form. Precede the short form with a single dash (`-`); precede the long forms with two dashes (`--`). Most options require argument values, except for boolean options, which must be present to switch a feature on. Options are not required for successful execution of the command.

If a subcommand requires a database name, and you do not specify one, `hadbm` will use the default database, `hadb`.

EXAMPLE 3–1 Example of hadbm command

The following illustrates the `status` subcommand:

```
hadbm status --nodes
```

Security Options

For security reasons, all `hadbm` commands require an administrator password. Use the `--adminpassword` option to set the password when you create a database or domain. From then on, you must specify that password when you perform operations on the database or domain.

For enhanced security, use the `--adminpasswordfile` option to specify a file containing the password, instead of entering it on the command line. Define the password in the password file with the following line:

`HADBM_ADMINPASSWORD=`*password*

Replace *password* with the password. Any other content in the file is ignored.

If you specify both the `--adminpassword` and `--adminpasswordfile` options, the `--adminpassword` takes precedence. If a password is required, but is not specified in the command, `hadbm` prompts you for a password.

Note – You can change the administrator password, if required, using the command `setadminpassword`. For more information about the command, see the man page for the command `setadminpassword`.

In addition to the administrator password, HADB also requires a database password to perform operations that modify the database schema. You must use both passwords when using the following commands: `hadbm create`, `hadbm addnodes`, and `hadbm refragment`.

Specify the database password on the command line with the `--dbpassword` option. Similar to the administrator password, you can also put the password in a file and use the `--dbpasswordfile` option, specifying the file location. Set the password in the password file with the following line:

`HADBM_DBPASSWORD=`*password*

For testing or evaluation, you can turn off password authentication with the `--no-adminauthentication` option when you create a database or domain. For more information, see "Creating a Database" on page 65 and "Creating a Management Domain" on page 64

The following table summarizes the `hadbm` security command line options.

TABLE 3–4 `hadbm` Security Options

Option (Short Form)	Description
--adminpassword=*password* -w	Specifies administrator password for the database or domain. If you use this option when you create a database or domain, then you must provide the password each time you use hadbm to operate on the database or domain. Use either this option or --adminpasswordfile, but not both.
--adminpasswordfile=*filepath* -W	Specifies file that contains the administrator password for the database or domain. If you use this option when you create a database or domain, then you must provide the password each time you use hadbm to operate on the database or domain. Use either this option or --adminpassword, but not both.
--no-adminauthentication -U	Use this option when you create a database or domain to specify that no administrator password is required. For security reasons, do not use this option in a production deployment.
--dbpassword= *password* -p	Specifies the database password. If you use this option when you create the database, then you must provide the password each time you use an `hadbm` command to operate on the database. Creates a password for the HADB system user. Must be at least 8 characters. Use either this option or --dbpasswordfile, but not both.
--dbpasswordfile= *filepath* -P	Specifies a file that contains the password for the HADB system user. Use either this option or --dbpassword, but not both.

General Options

General command options can be used with any `hadbm` subcommand. All are boolean options that are false by default. The following table describes the `hadbm` general command options.

TABLE 3–5 `hadbm` General Options

Option(Short Form)	Description
--quiet -q	Execute the subcommand silently without any descriptive messages.
--help -?	Display a brief description of this command and all the supported subcommands. No subcommand is required.
--version -V	Display the version details of the `hadbm` command. No subcommand is required.
--yes -y	Execute the subcommand in non-interactive mode.
--force -f	Execute the command non-interactively and does not throw an error if the command's post condition is already achieved.
--echo -e	Display the subcommand with all the options and their user-defined values or the default values, then executes the subcommand.
--agent=*URL* -m	URL to the management agents. *URL* is: *hostlist*:*port*, where *hostlist* is a comma separated list of hostnames or IP-addresses, and port is the port number on which the management agent is operating. Default is localhost:1862. NOTE: This option is not valid with `hadbm addnodes`.

Environment Variables

For convenience, you can set an environment variable instead of specifying a command option. The following table describes environment variables that correspond to `hadbm` command options.

TABLE 3–6 HADB Options and Environment Variables

Long Form	Short Form	Default	Environment Variable
--adminpassword	-w	none	$HADBM_ADMINPASSWORD
--agent	--m	localhost:1862	$HADBM_AGENT
--datadevices	-a	`1`	$HADBM_DATADEVICES
dbname	none	`hadb`	$HADBM_DB

TABLE 3–6 HADB Options and Environment Variables *(Continued)*

Long Form	Short Form	Default	Environment Variable
--dbpassword	-p	none	$HADBM_DBPASSWORD
--dbpasswordfile	-P	none	$HADBM_DBPASSWORDFILE
--devicepath	-d	Solaris and Linux: `/var/opt/SUNWhadb` Windows: `C:\Sun\AppServer\SUNWhadb\`*vers*, where *vers* is the HADB version number.	$HADBM_DEVICEPATH
--devicesize	-z	none	$HADBM_DEVICESIZE
--echo	-e	False	$HADBM_ECHO
--fast	-F	False	$HADBM_FAST
--force	-f	False	$HADBM_FORCE
--help	-?	False	$HADBM_HELP
--historypath	-t	Solaris and Linux: `/var/opt/SUNWhadb` Windows: REPLACEDIR, replaced by the actual URL at runtime.	$HADBM_HISTORYPATH
--hosts	-H	none	$HADBM_HOSTS
--interactive	-i	True	$HADBM_INTERACTIVE
--no-refragment	-r	False	$HADBM_NOREFRAGMENT
--portbase	-b	15200	$HADBM_PORTBASE
--quiet	-q	False	$HADBM_QUIET
--repair	-R	True	$HADBM_REPAIR
--rolling	-g	True	$HADBM_ROLLING
--saveto	-o	none	$HADBM_SAVETO
--set	-S	none	$HADBM_SET
--spares	-s	0	$HADBM_SPARES
--startlevel	-l	normal	$HADBM_STARTLEVEL
--version	-V	False	$HADBM_VERSION
--yes	-y	False	$HADBM_YES

Configuring HADB

This section describes the following basic HADB configuration tasks:

Creating a Management Domain

The command `hadbm createdomain` creates a management domain containing the specified HADB hosts. The command initializes internal communication channels between hosts and the persistence configuration store.

The syntax of the command is:

```
hadbm createdomain
 [--adminpassword=password |--adminpasswordfile=
file | --no-adminauthentication] [--agent=maurl]
 hostlist
```

The *hostlist* operand is a comma-separated list of HADB hosts, each of which is a valid IPv4 network address. Include all the hosts that you want to be in the new domain in the *hostlist*.

See "General Options" on page 61 for a description of the command options.

Before using this command, be sure an HADB management agent is running on every host in the *hostlist*. Additionally, the management agents must:

- Not be members of an existing domain.
- Be configured to use the same port.
- Be able to reach each other over UDP, TCP, and with IP multicast.

After `hadbm` creates the management domain, it enables all the hosts in the domain. Then the management agents are ready to manage databases. After creating HADB domains, the next step is to create the HADB database. For more information on creating HADB databases, see "Creating a Database" on page 65 .

EXAMPLE 3–2 Creating an HADB Management Domain

The following example creates a management domain on the four specified hosts:

`hadbm createdomain --adminpassword=` *password* `host1,host2,host3,host4`

After `hadbm` successfully executes the command, you will see the message:

```
Domain host1,host2,host3, host4 created.
```

EXAMPLE 3–2 Creating an HADB Management Domain *(Continued)*

After creating HADB domains, register the path and version of the HADB packages with the management agents.

Creating a Database

Use the `hadbm create` command to create a database manually.

Before you use this command to create a database, create the management domain and register the HADB package. If you have not performed these two steps when you run `hadbm create` , it implicitly performs them. Although this might seem like less work, failures in any of the commands can make debugging difficult. Besides, `hadbm create` is not atomic, that is, if one of the implicit commands fails, the commands that executed successfully will not be rolled back. Therefore, it is best to create the database only after creating the domain and registering the HADB package.

For example, if `hadbm createdomain` and `hadbm registerpackage` execute successfully but `hadbm create database` fails, the changes made by `hadbm createdomain` and `hadbm registerpackage` will persist.

▼ To create a database

1. **Create the management domain.**

   For more information, see "Creating a Management Domain" on page 64

2. **Register the HADB package.**

   For more information, see "Registering HADB Packages" on page 48 for more information.

3. **Use the hadbm create command to create the database.**

   For information on command syntax, see the following section.

`hadbm create` Command Syntax

`hadbm create` **[**`--package=`*name*`]` **[**`--packagepath=`*path*`]` **[**`--historypath=`*path*`]** **[**`--devicepath=`*path*`]` **[**`--datadevices=`*number* **]** **[**`--portbase=`*number*`]** **[**`--spares=`*number*`]` **[**`--set=`*attr-val-list*`]` **[**`--agent=`*maurl*`]` **[**`--no-cleanup`**]** **[** `--no-clear` **]** **[** `--devicesize =`*size*`]` **[**`--dbpassword=`*password* **|** `--dbpasswordfile=`*file* **]** `--hosts=`*host list* **[**`--adminpassword=`*password* **|** `--adminpasswordfile=`*file* **|** `--no-adminauthentication` **]** [*dbname*]

The *dbname* operand specifies the database name, which must be unique. To make sure the database name is unique, use the `hadbm list` command to list existing database names. Use the

default database name unless you need to create multiple databases. For example, to create multiple clusters with independent databases on the same set of HADB machines, use a separate database name for each cluster.

The `hadbm create` command writes error messages to the console, not log files.

Table 3–7 describes the special `hadbm create` command options. See "General Options" on page 61 for a description of additional command options.

TABLE 3–7 hadbm create Options

Option(Short Form)	Description	Default
--datadevices= *number* -a	Number of data devices on each node, between one and eight inclusive. Data devices are numbered starting at 0.	1
--devicepath= *path* -d	Path to the devices. There are four devices: ■ `DataDevice` ■ `NiLogDevice` (node internal log device) ■ `RelalgDevice` (relational algebra query device) ■ `NoManDevice` (node manager device). This path must exist and be writable. To set this path differently for each node or each device, see "Setting Heterogeneous Device Paths" on page 68	Solaris and Linux: `/var/opt/SUNWhadb` Windows: `C:\Sun\AppServer\SUNWhadb\`*vers*, where *vers* is the HADB version number. Default is specified by `ma.server.dbdevicepath in` management agent configuration file. For more details, see "Configuration File" on page 53
--devicesize= *size* -z	Device size for each node. For more information, see "Specifying Device Size" on page 68. Increase the device size as described in "Adding Storage Space to Existing Nodes" on page 87	1024MB Maximum size is lesser of maximum operating system file size or 256 GB. Minimum size is: (4 x `LogbufferSize` + 16MB) / *n* Where *n* is the number of data devices given by the option `--datadevices`.
--historypath= *path* -t	Path to the history files. This path must already exist and be writable. For more information on history files, see "Clearing and Archiving History Files" on page 101	Default is specified by `ma.server.dbhistorypath` in management agent configuration file. For details, see "Configuration File" on page 53 Solaris and Linux:`/var/opt/SUNWhadb` On Windows: REPLACEDIR (replaced by the actual URL at runtime.)

TABLE 3–7 hadbm create Options *(Continued)*

Option(Short Form)	Description	Default
--hosts= *hostlist* -H	Comma-separated list of host names or IP addresses (IPv4 only) for the nodes in the database. Use IP addresses to avoid dependence on DNS lookups. Host names must be absolute. You cannot use `localhost` or `127.0.0.1` as a host name. See "Specifying Hosts" on page 68 for more information.	None
--package=*name* -k	Name of the HADB package (version). If the package is not found, a default package is registered. This option is deprecated. Use the hadbm registerpackage command to register a package in the domain.	None
--packagepath=*path*-L	Path to the HADB software package. Use only if the package is not registered in the domain. This option is deprecated. Use the hadbm registerpackage command to register a package in the domain.	None
--portbase= *number* -b	Port base number used for node 0. Successive nodes are automatically assigned port base numbers in steps of 20 from this number. Each node uses its port base number and the next five consecutively numbered ports. To run several databases on the same machine, have a plan for allocating port numbers explicitly.	15200
--spares= *number* -s	Number of spare nodes. This number must be even and must be less than the number of nodes specified in the `--hosts` option.	0
--set=*attr-val-list* -S	Comma-separated list of database configuration attributes in *name =value* format. For explanations of database configuration attributes, see "Clearing and Archiving History Files" on page 101	None

EXAMPLE 3–3 Example of creating a database

The following command is an example of creating a database:

EXAMPLE 3–3 Example of creating a database *(Continued)*

```
hadbm create --spares 2 --devicesize 1024 --dbpassword secret123
 --hosts n0,n1,n2,n3,n4,n5
```

Specifying Hosts

Use the `--hosts` option to specify a comma-separated list of host names or IP addresses for the nodes in the database. The `hadbm create` command creates one node for each host name (or IP address) in the list. The number of nodes must be even. Use duplicate host names to create multiple nodes on the same machine with different port numbers. Make sure that nodes on the same machine are not mirror nodes.

Nodes are numbered starting at zero in the order listed in this option. The first mirrored pair are nodes zero (0) and one (1), the second two (2) and three (3), and so on. Odd numbered nodes are in one DRU, even numbered nodes in the other. With `--spares` option, spare nodes are those with the highest numbers.

For information about configuring double network interfaces, see "Configuring Network Redundancy" on page 35

Specifying Device Size

Specify the device size using the `--devicesize` option. The recommended device size is:

(4x / nd + 4l/d) / 0.99

Where

- *x* is the total size of user data
- *n* is the number of nodes (given by the `--hosts` option)
- *d* is the number of devices per node (given by the `--datadevices` option)
- *l* is the log buffer size (given by the attribute `LogBufferSize`)

  If refragmentation might occur (for example, using `hadbm addnodes`), then the recommended device size is:

  (8x / nd + 4l/d) / 0.99

Setting Heterogeneous Device Paths

To set a different device path for each node or service, use the `-- set` option of `hadbm create`. There are four types of devices: the `DataDevice`, the `NiLogDevice` (node internal log device), the `RelalgDevice` (relational algebra query device), and the `NoManDevice` (node manager device). The syntax for each *name =value* pair is as follows, where *-devno* is required only if the *device* is `DataDevice`:

```
node-nodeno.device-devno.Devicepath
```

For example:

```
--set Node-0.DataDevice-0.DevicePath=/disk0,
Node-1.DataDevice-0.DevicePath=/disk 1
```

You can also set a heterogeneous path to history files, as follows:

`node-`*`nodeno`*`.historypath=`*`path`*

For information on history files, see "Clearing and Archiving History Files" on page 101

Any device path that is not set for a particular node or device defaults to the `--devicepath` value.

Note – Change device paths and location of history files using `hadbm set` and `hadbm addnodes` commands.

Troubleshooting

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

- Ensure you have started the management agents on all the hosts and defined an HADB domain. For details, see "Starting the Management Agent" on page 54
- File and directory permissions must be set to allow read, write, and execute access to the install, history, device, and config paths for the following users:
  - Sun Java System Application Server administrative user (set during installation)
  - HADB system user

    For details about setting user permissions, see "Preparing for HADB Setup" on page 34

  Application Server and HADB port assignments must not conflict with other port assignments on the same machine. Default recommended port assignments are:

  - Sun Java SystemMessage Queue: 7676
  - IIOP: 3700
  - HTTP server: 80
  - Administration server: 4848
  - HADB nodes: Each node uses six consecutive ports. For example, for default port 15200, node 0 uses 15200 through 15205 , node 1 uses 15220 through 15225, and so on.

  Disk space must be adequate; see the *Sun Java System Application Server Release Notes*.

Viewing and Modifying Configuration Attributes

You can view and modify database configuration attributes with the `hadbm get` and `hadbm set` commands, respectively.

Getting the Values of Configuration Attributes

To get the values of configuration attributes, use the `hadbm get` command. For a list of valid attributes, see "Configuration Attributes" on page 71. The command syntax is:

```
hadbm get attribute-list | --all
[dbname]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

The *dbname* operand specifies the database name. The default is `hadb`.

The *attribute-list* operand is a comma-separated or quote-enclosed space-separated list of attributes. The `--all` option displays values for all attributes. For a list of all attributes for `hadbm get`, see "Configuration Attributes" on page 71.

See "General Options" on page 61 for a description of command options.

EXAMPLE 3–4 Example of using hadbm get

```
hadbm get JdbcUrl,NumberOfSessions
```

Setting the Values of Configuration Attributes

To set the values of configuration attributes, use the `hadbm set` command. For a list of valid attributes, see "Configuration Attributes" on page 71

```
hadbm set [dbname] attribute
=value[,attribute=
value...]
 [--adminpassword=password | --adminpasswordfile=file]
 [--agent=maurl]
```

The *dbname* operand specifies the database name. The default is `hadb`.

The *attribute=value* list is a comma-separated or quote-enclosed space-separated list of attributes.

See "General Options" on page 61 for a description of command options.

If this command executes successfully, it restarts the database in the state it was in previously, or in a better state. For information about database states, see "Getting the Status of HADB" on page 92 restart the HADB as described in "Restarting a Database" on page 83.

You cannot set the following attributes with `hadbm set`. Instead, set them when you create a database (see "Creating a Database" on page 65).

- DatabaseName
- DevicePath
- HistoryPath
- NumberOfDatadevices
- Portbase
- JdbcUrl (its value is set during database creation based on the `--hosts` and `--portbase` options).

Note – Using `hadbm set` to set any configuration attribute, except `ConnectionTrace` or `SQLTraceMode`, causes a rolling restart of HADB. In a rolling restart, each node is stopped, and started with the new configuration, one at a time; HADB services are not interrupted.

If you set `ConnectionTrace` or `SQLTraceMode`, no rolling restart occurs, but the change only takes effect for new HADB connections made from an Application Server instance.

Configuration Attributes

The following table lists the configuration attributes that you can modify with `hadbm set` and retrieve with `hadbm get`.

TABLE 3–8 Configuration Attributes

Attribute	Description	Default	Range
ConnectionTrace	If true, records a message in the HADB history files when a client connection (JDBC, ODBC) is initiated or terminated.	False	True or False
CoreFile	Do not change the default value.	False	True or False
DatabaseName	Name of the database.	hadb	
DataBufferPoolSize	Size of the data buffer pool allocated in shared memory.	200MB	16 - 2047 MB

TABLE 3–8 Configuration Attributes *(Continued)*

Attribute	Description	Default	Range
DataDeviceSize	Specifies the device size for the node. For information on the recommended `DataDeviceSize`, see "Specifying Device Size" on page 68 The maximum value is the smaller of 256GB or the maximum operating system file size. The minimum value is: (4 x LogbufferSize + 16MB) / *n* where *n* is number of data devices.	1024MB	32 - 262144 MB
PackageName	Name of HADB software package used by the database.	V4.x.x.x	None
DevicePath	Location of the devices. Devices are: ■ Data device (`DataDevice`) ■ Node internal log device (`NiLogDevice`) ■ Relational algebra query device (`RelalgDevice`)	Solaris and Linux: `/var/opt/SUNWhadb` Windows: `C:\Sun\AppServer\SUNWhadb\`*vers*, where *vers* is the HADB version number.	
EagerSessionThreshold	Determines whether normal or eager idle session expiration is used. In normal idle session expiration, sessions that are idle for more than `SessionTimeout` seconds are expired. When the number of concurrent sessions exceeds the `EagerSessionThreshold` percentage of the maximum number of sessions, sessions that are idle for more than `EagerSessionTimeout` seconds are expired.	Half of NumberOfSessions attribute	0 - 100
EagerSessionTimeout	The time in seconds a database connection can be idle before it expires when eager session expiration is used.	120 seconds	0-2147483647 seconds

TABLE 3–8 Configuration Attributes *(Continued)*

Attribute	Description	Default	Range
EventBufferSize	Size of the event buffer, where database events are logged. If set to `0`, no event buffer logging is performed. During failures, the event buffer is dumped. This gives valuable information on the cause of the failures and is useful during trial deployment. Writing events to memory has a performance penalty.	`0 MB`	0-2097152 MB
HistoryPath	Location of the HADB history files, which contain information, warnings, and error messages. This is a read-only attribute.	Solaris and Linux: `/var/opt/SUNWhadb` Windows: REPLACEDIR (replaced by the actual URL at runtime.)	
InternalLogbufferSize	Size of the node internal log device, which keeps track of operations related to storing data.	12MB	4 - 128 MB
JdbcUrl	The JDBC connection URL for the database. This is a read-only attribute.	none	
LogbufferSize	Size of the log buffer, which keeps track of operations related to data.	48MB	4 - 2048 MB
MaxTables	Maximum number of tables allowed in an HADB database.	1100	100 - 1100
NumberOfDatadevices	Number of data devices used by an HADB node. This is a read-only attribute.	1	1 - 8
NumberOfLocks	Number of locks allocated by an HADB node.	50000	20000-1073741824
NumberOfSessions	Maximum number of sessions (database connections) that can be opened for an HADB node.	100	1 - 10000
PortBase	Base port number used to create different port numbers for different HADB processes. This is a read-only attribute.	15200	10000 - 63000
RelalgDeviceSize	Size of the device used in relational algebra queries.	128 MB	32 - 262144 MB

TABLE 3–8 Configuration Attributes *(Continued)*

Attribute	Description	Default	Range
SessionTimeout	Amount of time a database connection can be idle before it expires when normal session expiration is used.	1800 seconds	0-2147483647 seconds
SQLTraceMode	Amount of information about executed SQL queries written to the history files. If `SHORT`, login and logout of SQL sessions are recorded. If `FULL`, all SQL queries being prepared and executed, including parameter values, are recorded.	NONE	NONE/SHORT/FULL
StartRepairDelay	Maximum time a spare node allows for a failed active node to perform a node recovery. If the failed node cannot recover within this time interval, the spare node starts copying data from the failed node's mirror and becomes active. Changing the default value is not recommended.	20 seconds	0 - 100000 seconds
StatInterval	Interval at which an HADB node writes throughput and response time statistics to its history file. To disable, set to 0. Here is an example of a statistics line: `Req-reply time: # 123, min= 69 avg= 1160 max= 9311 %=100.0` The number after the has sign (`#`) is the number of requests serviced over the `StatInterval`. The next three numbers are the minimum, average, and maximum time in microseconds taken by transactions completed over the `StatInterval`. The number afer the percent sign (`%`) is the number of transactions completed successfully within 15 milliseconds over the `StatInterval`.	600 seconds	0 - 600 seconds
SyslogFacility	Facility used when reporting to `syslog`. The `syslog` daemon should be configured (see `man syslogd.conf` for details). Use a facility that is not used by other applications running on the same machine. Set to `none` to disable `syslog` logging.	local0	local0, local1, local2, local3, local4, local5, local6, local7, kern, user, mail, daemon, auth, syslog, lpr, news, uucp, cron, none
SysLogging	If true, an HADB node writes information to the operating system's `syslog` files.	True	True or False

TABLE 3–8 Configuration Attributes *(Continued)*

Attribute	Description	Default	Range
SysLogLevel	Minimum level of HADB message saved to operating system's `syslog` files. All messages of that level or higher will be logged. For example, "info" logs all messages.	warning	nonealert errorwarninginfo
SyslogPrefix	Text string inserted before all `syslog` messages written by the HADB.	hadb -dbname	
TakeoverTime	Time between when a node fails and when its mirror takes over. Do not change the default value.	10000 (milliseconds)	500 - 16000 milliseconds

Configuring the JDBC Connection Pool

Application Server communicates with HADB using the Java Database Connectivity (JDBC) API. The `asadmin configure-ha-cluster` command automatically creates a JDBC connection pool for use with HADB (for a cluster *cluster-name*). The name of the connection pool is *cluster-name*`-hadb-pool`. The JNDI URL of JDBC resource is `jdbc/`*cluster-name*`-hastore`.

The initial configuration of the connection pool is normally sufficient. When you add a node, change the steady pool size so that there are eight connections for each HADB active node. See "Adding Nodes" on page 88.

This chapter covers the following topics:

- "Getting the JDBC URL" on page 75
- "Creating a Connection Pool" on page 76
- Example 3–5
- "Creating a JDBC Resource" on page 77

  For general information about connection pools and JDBC resources, see *Administration Guide*.

Getting the JDBC URL

Before you can set up the JDBC connection pool, you need to determine the JDBC URL of HADB using the `hadbm get` command as follows:

```
hadbm get JdbcUrl [dbname]
```

For example:

```
hadbm get JdbcUrl
```

This command displays the JDBC URL, which is of he following form:

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

Remove the `jdbc:sun:hadb:` prefix and use the *host*`:`*port*`,` *host*`:`*port*`...` part as the value of the `serverList` connection pool property, described in Table 3–10.

Creating a Connection Pool

The following table summarizes connection pool settings required for the HADB. Change the Steady Pool Size when adding nodes, but do not change other settings.

TABLE 3–9 HADB Connection Pool Settings

Setting	Required Value for HADB
Name	The HADB JDBC resource's Pool Name setting must refer to this name
Database Vendor	HADB 4.4
Global Transaction Support	Unchecked/false
DataSource Classname	`com.sun.hadb.jdbc.ds.HadbDataSource`
Steady Pool Size	Use 8 connections for each active HADB node. For more detailed information, see the *System Deployment Guide*.
Connection Validation Required	Checked/true
Validation Method	`meta-data`
Table Name	Do not specify
Fail All Connections	Unchecked/false
Transaction Isolation	`repeatable-read`
Guarantee Isolation Level	Checked/true

The following table summarizes connection pool properties required for the HADB. Change `serverList` when adding nodes, but do not change other properties.

TABLE 3–10 HADB Connection Pool Properties

Property	Description
username	Name of the `storeuser` to use in the `asadmin create-session-store` command.
password	Password (`storepassword`) to us in the `asadmin create-session-store` command.

TABLE 3–10 HADB Connection Pool Properties *(Continued)*

Property	Description
serverList	JDBC URL of the HADB. To determine this value, see "Getting the JDBC URL" on page 75 You must change this value if you add nodes to the database. See "Adding Nodes" on page 88.
cacheDatabaseMetaData	When `false` , as required, ensures that calls to `Connection.getMetaData()` make calls to the database, which ensures that the connection is valid.
eliminateRedundantEndTransaction	When `true` , as required, improves performance by eliminating redundant commit and rollback requests and ignoring these requests if no transaction is open.
maxStatement	Maximum number of statements per open connection that are cached in the driver statement pool. Set this property to `20`.

EXAMPLE 3–5 Creating a Connection Pool

Here is an example `asadmin create-jdbc-connection-pool` command that creates an HADB JDBC connection pool:

```
asadmin create-jdbc-connection-pool
--user adminname --password secret
--datasourceclassname com.sun.hadb.jdbc.ds.HadbDataSource
--steadypoolsize=32
--isolationlevel=repeatable-read
--isconnectvalidatereq=true
--validationmethod=meta-data
--property username=storename:password=secret456:serverList=
host\:port,host\:port,
host\\:port,host\:port,
host\:port,host\:port
:cacheDatabaseMetaData=false:eliminateRedundantEndTransaction=true hadbpool
```

On Solaris, escape colon characters (:) within property values with double backslashes (\\ ). On Windows, escape colon characters (:) with single backslashes (\ ).

Creating a JDBC Resource

The following table summarizes JDBC resource settings required for HADB.

TABLE 3–11 HADB JDBC Resource Settings

Setting	Description
JNDI Name	The following JNDI name is the default in the session persistence configuration: `jdbc/hastore`. You can use the default name or a different name. You must also specify this JNDI name as the value of the `store-pool-jndi-name` Persistence Store property when you activate the availability service.
Pool Name	Select from the list the name (or ID) of the HADB connection pool used by this JDBC resource. For more information, see "Configuring Network Redundancy" on page 35
Data Source Enabled	Checked/true

Managing HADB

You generally need to perform management operations when you replace or upgrade your network, hardware, operating system, or HADB software. The following sections explain various management operations:

Managing Domains

You can perform the following operations on an HADB domain:

See "Security Options" on page 60 and "General Options" on page 61 for a description of command options.

Extending a Domain

Use `extenddomain` to add hosts to an existing management domain. The command syntax is:

```
hadbm extenddomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
hostlist
```

IP addresses of HADB hosts must be IPv4 addresses.

For more information, see `hadbm-extenddomain`(1).

Deleting a Domain

Use `deletedomain` to remove a management domain. The command syntax is:

```
hadbm deletedomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

For more information, see `hadbm-deletedomain`(1).

Removing Hosts from a Domain

Use `reducedomain` to remove hosts from the management domain. The command syntax is:

```
hadbm reducedomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
host_list
```

For more information, see `hadbm-reducedomain`(1).

Listing Hosts in a Domain

Use `listdomain` to list all hosts defined in the management domain. The command syntax is:

```
hadbm listdomain
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

For more information, see `hadbm-listdomain`(1).

Managing Nodes

You can perform the following operations on individual nodes:

- “Starting a Node” on page 80
- “Stopping a Node” on page 81
- “Restarting a Node” on page 81

Starting a Node

You might need to manually start an HADB node that was stopped because its host was taken off-line for a hardware or software upgrade or replacement. Also, you might need to manually start a node if it fails to restart for some reason (other than a double failure). For more information on how to recover from double failures, see "Clearing a database" on page 84.

In most cases, you should first attempt to start the node using the `normal` start level. You must use the `repair` start level if the `normal` start level fails or times out.

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

```
hadbm startnode
 [--adminpassword=password | --adminpasswordfile=file]
 [--agent=maurl]
 [--startlevel=level]
 nodeno
 [dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

The *nodeno* operand specifies the number of the node to start. Use `hadbm status` to display the numbers of all nodes in a database.

For more information, see `hadbm-startnode`(1).

Start level option

The `hadbm startnode` command has one special option, `--startlevel` (short form `-l`), that specifies the level at which to start the node.

Node start levels are:

- **normal** (default): starts the node with the data found locally on the node (in the memory and in the data device file on the disk) and synchronizes it with the mirror for recent updates it missed.
- **repair**: forces the node to discard local data and copy it from its mirror.
- **clear**: reinitializes the devices for the node and forces a repair of data from its mirror node. Use when the device files need to be initialized, necessary if they are damaged or the disk that contained the device files is replaced.

See "General Options" on page 61 for a description of other command options.

EXAMPLE 3–6 Example of starting a node

```
hadbm startnode 1
```

Stopping a Node

You might need to stop a node to repair or upgrade the host machine's hardware or software. To stop a node, use the `hadbm stopnode` command. The command syntax is:

```
hadbm stopnode
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[--no-repair]
nodeno
[dbname]
```

The *nodeno* operand specifies the number of the node to stop. The mirror node of this node number must be running. Use `hadbm status` to display the numbers of all nodes in a database.

The *dbname* operand specifies the database name. The default is `hadb`.

The `hadbm stopnode` command has one special option, `--no-repair` (short form `-R`) that indicates no spare node is to replace the stopped node. Without this option, a spare node starts up and takes over the functioning of the stopped node.

See "General Options" on page 61 for a description of other command options. For more information, see `hadbm-stopnode`(1).

EXAMPLE 3–7 Example of stopping a node

```
hadbm stopnode 1
```

Restarting a Node

You might have to restart a node if you notice unusual behavior such as excessive CPU consumption.

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

```
hadbm restartnode
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[--startlevel=level]
nodeno
[dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

The *nodeno* operand specifies the number of the node to restart. Use `hadbm status` to display the numbers of all nodes in a database.

The `hadbm restartnode` command has one special option, `--startlevel` (short form `-l`), that specifies the level at which to start the node. See "Start level option" on page 80 for more information.

See "General Options" on page 61 for a description of other command options. For more information, see `hadbm-restartnode`(1).

EXAMPLE 3–8 Example of restarting a node

```
hadbm restartnode 1
```

Managing Databases

You can perform the following operations on HADB databases:

Starting a Database

To start a database, use the `hadbm start` command. This command starts all nodes that were running before the database was stopped. Individually stopped (offline) nodes are not started when the database is started after a stop.

The command syntax is:

```
hadbm start
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

See "General Options" on page 61 for a description of command options. For more information, see `hadbm-start`(1).

EXAMPLE 3–9 Example of starting a database

```
hadbm start
```

Stopping a Database

When you stop and start a database in separate operations, data is unavailable while it is stopped. To keep data available, you can restart a database as described in "Restarting a Database" on page 83.

Stop a database to:

- Remove the database.
- Perform system maintenance that affects all HADB nodes.

Before stopping a database, either stop dependent Application Server instances that are using the database, or configure them to use a Persistence Type other than `ha`.

When you stop the database, all the running nodes in the database are stopped and the status of the database becomes Stopped. For more information about database states, see "Getting the Status of HADB" on page 92.

To stop a database, use the `hadbm stop` command. The command syntax is:

```
hadbm stop
[--adminpassword=password | --adminpasswordfile= file]
[--agent=maurl]
[dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

See "General Options" on page 61 for a description of command options. For more information, see `hadbm-stop`(1).

EXAMPLE 3–10 Example of stopping a database

```
hadbm stop
```

Restarting a Database

You might want to restart a database if you notice strange behavior (for example consistent timeout problems). In some cases, a restart may solve the problem.

When you restart a database, y, the database and its data remain available. When you stop and start HADB in separate operations, data and database services are unavailable while HADB is stopped. This is because by default `hadbm restart` performs a rolling restart of nodes: it stops and starts the nodes one by one. In contrast, `hadbm stop` stops all nodes simultaneously.

To restart a database, use the `hadbm restart` command. The command syntax is:

```
hadbm restart
[--adminpassword=password | --adminpasswordfile=file]
```

```
[--agent=maurl]
[--no-rolling]
[dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

This command has one special option, `--no-rolling` (short form `-g`), that specifies to restart all nodes at once, which causes loss of service. Without this option, this command restarts each of the nodes in the database to the current state or a better state.

See "General Options" on page 61 for a description of other command options. For more information, see `hadbm-restart`(1).

For example:

```
hadbm restart
```

Listing Databases

To list all the databases in an HADB instance, use the `hadbm list` command. The command syntax is:

```
hadbm list
[--agent=maurl]
[--adminpassword=password | --adminpasswordfile=file]
```

See "General Options" on page 61 for a description of command options. For more information, see `hadbm-list`(1).

Clearing a database

Clear a database when:

- The `hadbm status` command reveals that the database is non-operational or if See "Getting the Status of HADB" on page 92.
- Multiple nodes do not respond and are in waiting state for a long time.
- Recovering from session data corruption. See "Recovering from Session Data Corruption" on page 86

The `hadbm clear` command stops the database nodes, clears the database devices, then starts the nodes. This command erases the Application Server schema data store in HADB, including tables, user names, and passwords. After running `hadbm clear`, use `asadmin configure-ha-cluster` to recreate the data schema, reconfigure the JDBC connection pool, and reload the session persistence store.

The command syntax is:

```
hadbm clear  [--fast]  [--spares=number]
[--dbpassword=password | --dbpasswordfile= file]
[--adminpassword=password | --adminpasswordfile= file]
[--agent=maurl]
[dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

The following table describes the special `hadbm clear` command options. See "General Options" on page 61 for a description of other options.

For more information, see `hadbm-clear`(1).

TABLE 3–12 `hadbm clear` Options

Option	Description	Default
--fast -F	Skips device initialization while initializing the database. Do not use if the disk storage device is corrupted.	Not applicable
--spares= *number* -s	Number of spare nodes the reinitialized database will have. Must be even and less than the number of nodes in the database.	Previous number of spares

For example:

```
hadbm clear --fast --spares=2 --dbpassword secret123
```

Removing a Database

To remove an existing database, use the `hadbm delete` command. This command deletes the database's configuration files, device files, and history files, and frees shared memory resources. The database you want to remove must exist and must be stopped. See "Stopping a Database" on page 83.

The command syntax is:

```
hadbm delete
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

See "General Options" on page 61 for a description of command options. For more information, see `hadbm-delete`(1).

EXAMPLE 3–11 Example of removing a database

The command:

```
hadbm delete
```

deletes the default database, `hadb`.

Recovering from Session Data Corruption

The following are indications that session data may be corrupted:

- Error messages appear in the Application Server system log (`server.log`) every time an application tries to save session state.
- Error messages in the server log indicate that the session could not be found or could not be loaded during session activation.
- Sessions that are activated after previously being passivated contain empty or incorrect session data.
- When an instance fails, failed-over sessions contain empty or incorrect session data.
- When an instance fails, instances that try to load a failed-over session cause an error in the server log indicating the session could not be found or could not be loaded.

▼ To bring the session store back to a consistent state

If you determine that the session store has been corrupted, bring it back to a consistent state by following this procedure:

1. **Clear the session store.**

   Determine if this action corrects the problem. If it does, then stop. If not—for example, if you continue to see errors in the server log—then continue.

2. **Re-initialize the data space on all the nodes and clear the data in the database.**

   See "Clearing a database" on page 84 .

   Determine if this action corrects the problem. If it does, then stop. If not—for example, if you continue to see errors in the server log—then continue.

3. **Delete and then recreate the database.**

   See "Removing a Database" on page 85 and "Creating a Database" on page 65

Expanding HADB

There are two reasons to expand your original HADB configuration:

- Volume of session data being saved increases beyond existing storage space in data devices. Transactions may start aborting due to full data devices.
- User load increases, exhausting system resources. You need to add more hosts.

This section describes how you can expand HADB without shutting down your Application Server cluster or database, in particular:

Also see related information in "Maintaining HADB Machines" on page 99 .

Adding Storage Space to Existing Nodes

Add HADB storage space:

- If user transactions repeatedly abort with one of the following error messages:
  - `4592: No free blocks on data devices`
  - `4593: No unreserved blocks on data devices`
- If the `hadbm deviceinfo` command consistently reports insufficient free size. See "Getting Device Information" on page 94.

You may also want to add storage space to existing nodes if there is unused disk space on the nodes or when you add disk capacity. For information on the recommended data device size, see "Specifying Device Size" on page 68

To add storage space to nodes, use the `hadbm set` command to increase data device size.

The command syntax is:

```
hadbm set DataDeviceSize=size
```

where *size* is the data device size in MBytes.

See "General Options" on page 61 for a description of command options.

Changing the data device size for a database in a `FaultTolerant` or higher state upgrades the system without loss of data or availability. The database remains in operational during the reconfiguration. Changing device size on a system that is not `FaultTolerant` or better causes loss of data. For more information about database states, see "Database States" on page 93.

EXAMPLE 3–12 Example of setting data device size

The following command is an example of setting data device size:

```
hadbm set DataDeviceSize=1024
```

Adding Machines

You may want to add machines if HADB requires more processing or storage capacity. To add a new machine on which to run HADB, install HADB packages with or without the Application Serveras described in Chapter 2, "Installing and Setting Up High Availability Database." For an explanation of node topology alternatives, see Chapter 3, "Selecting a Topology," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide*.

▼ To add new machines to an existing HADB instance

1. **Start management agents on the new nodes.**
2. **Extend the management domain to the new hosts.**
   For details, see `hadbm extenddomain` command.
3. **Start the new nodes on these hosts.**
   For details, see "Adding Nodes" on page 88

Adding Nodes

To increase processing and storage capacity of an HADB system, create new nodes and add them to the database.

After you add nodes, update the following properties of the HADB JDBC connection pool:

- The `serverlist` property.
- Steady pool size. Generally, you add 8 more connections for each new node. For more information, see "System Sizing" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide*.

To add nodes, use the `hadbm addnodes` command. The command syntax is:

```
hadbm addnodes  [--no-refragment]  [--spares=sparecount]
[--historypath=path]
[--devicepath=path]
[--set=attr-name-value-list]
[--dbpassword=password | --dbpasswordfile=file ]
```

```
[--adminpassword=password | --adminpasswordfile=file]
--hosts=hostlist  [dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`. The database must be in `HAFaultTolerant` or `FaultTolerant` state. For more information about database states, see "Getting the Status of HADB" on page 92.

If you do not specify the `--devicepath` and `--historypath` options, the new nodes will have the same device path and use the same history files as the existing database.

Adding nodes performs a refragmentation and redistribution of the existing data to include the new nodes in the system. Online refragmenting requires that the disks for the HADB nodes have enough space to contain the old data and the new data simultaneously until refragmenting is finished, that is, the user data size must not exceed 50% of the space available for user data. For details, see "Getting Device Information" on page 94

Note – The best time to add nodes is when the system is lightly loaded.

EXAMPLE 3–13 Example of adding nodes

For example:

```
hadbm addnodes --dbpassword secret123 -adminpassword=
password --hosts n6,n7,n8,n9
```

The following table describes the special `hadbm addnodes` command options. See "General Options" on page 61 for a description of other options.

TABLE 3–13 `hadbm addnodes` Options

Option	Description	Default
--no-refragment -r	Do not refragment the database during node creation; In this case, refragment the database later using the `hadbm refragment` command to use the new nodes. For details about refragmentation, see "Refragmenting the Database" on page 90 If you do not have sufficient device space for refragmentation, recreate the database with more nodes. See "Adding Nodes by Recreating the Database" on page 91	Not applicable
--spares= *number* -s	Number of new spare nodes in addition to those that already exist. Must be even and not greater than the number of nodes added.	0

TABLE 3–13 `hadbm addnodes` Options *(Continued)*

Option	Description	Default
--devicepath= *path* -d	Path to the devices. Devices are: ■ `DataDevice` ■ `NiLogDevice` (node internal log device) ■ `RelalgDevice` (relational algebra query device) This path must already exist and be writable. To set this path differently for each node or each device, see "Setting Heterogeneous Device Paths" on page 68	Solaris and Linux: *HADB_install_dir*`/device` Windows: `C:\Sun\AppServer\SUNWhadb\`*vers*, where *vers* is the HADB version number.
--hosts= *hostlist* -H	Comma-separated list of new host names for the new nodes in the database. One node is created for each comma-separated item in the list. The number of nodes must be even. IP addresses of HADB hosts must be IPv4 addresses. Using duplicate host names creates multiple nodes on the same machine with different port numbers. Make sure that nodes on the same machine are not mirror nodes. Odd numbered nodes are in one DRU, even numbered nodes in the other. If `--spares` is used, new spare nodes are those with the highest numbers. If the database was created with double network interfaces, the new nodes must be configured in the same way. See "Configuring Network Redundancy" on page 35 .	None

Refragmenting the Database

Refragment the database to store data in newly-created nodes. Refragmentation distributes data evenly across all active nodes.

To refragment the database, use the `hadbm refragment` command. The command syntax is:

```
hadbm refragment  [--dbpassword=password | --dbpasswordfile=file]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`. The database must be in `HAFaultTolerant` or `FaultTolerant` state. For more information about database states, see "Getting the Status of HADB" on page 92.

See "General Options" on page 61 for a description of command options. For more information, see `hadbm-refragment`(1).

Online refragmentation requires that the disks for the HADB nodes have enough space to contain the old data and the new data simultaneously until refragmenting is finished, that is, the user data size must not exceed 50% of the space available for user data. For details, see "Getting Device Information" on page 94

Note – The best time to refragment the database is when the system is lightly loaded.

If this command fails after multiple attempts, see "Adding Nodes by Recreating the Database" on page 91

EXAMPLE 3–14 Example of refragmenting the database

For example:

```
hadbm refragment --dbpassword secret123
```

Adding Nodes by Recreating the Database

If online refragmentation fails persistently when you add new nodes (either due to lack of data device space or other reasons), recreate the database with new nodes. This will lead to the loss of existing user data and schema data.

▼ To add nodes by recreating the database

This procedure enables you to maintain HADB availability throughout the process.

1. **For each Application Server instance:**

   a. **Disable the Application Server instance in the load balancer.**

   b. **Disable session persistence.**

   c. **Restart the Application Server instance.**

   d. **Re-enable the Application Server instance in the load balancer.**

   If you do not need to maintain availability, you can disable and re-enable all the server instances at once in the load balancer. This saves time and prevents failover of outdated session data.

2. **Stop the database as described in "Stopping a Database" on page 83 .**

3. **Delete the database as described in "Removing a Database" on page 85 .**

4. **Recreate the database with the additional nodes as described in "Creating a Database" on page 65**

5. **Reconfigure the JDBC connection pool as described in "Configuring the JDBC Connection Pool" on page 75**

6. **Reload the session persistence store.**

7. **For each Application Server instance:**

   a. **Disable the Application Server instance in the load balancer.**

   b. **Enable session persistence.**

   c. **Restart the Application Server instance.**

   d. **Re-enable the Application Server instance in the load balancer.**

   If you do not need to maintain availability, you can disable and re-enable all the server instances at once in the load balancer. This saves time and prevents failover of outdated session data.

Monitoring HADB

You can monitor the activities of HADB by:

- "Getting the Status of HADB" on page 92
- "Getting Device Information" on page 94
- "Getting Runtime Resource Information" on page 96

These sections briefly describe the `hadbm status`, `hadbm deviceinfo`, and `hadbm resourceinfo` commands. For information on interpreting HADB information, see "Performance" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Performance Tuning Guide*.

Getting the Status of HADB

Use the `hadbm status` command to display the status of the database or its nodes. The command syntax is:

```
hadbm status
[--nodes]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

The `--nodes` option (short form `-n`) displays information on each node in the database. For more information , see "Node Status" on page 93. See "General Options" on page 61 for a description of other command options.

For more information, see `hadbm-status`(1).

EXAMPLE 3–15 Example of getting HADB status

For example:

```
hadbm status --nodes
```

Database States

A database's *state* summarizes its current condition. The following table describes the possible database states.

TABLE 3–14 HADB States

Database State	Description
High-Availability Fault Tolerant (HAFaultTolerant)	Database is fault tolerant and has at least one spare node on each DRU.
Fault Tolerant	All the mirrored node pairs are up and running.
Operational	At least one node in each mirrored node pair is running.
Non Operational	One or more mirrored node pairs is missing both nodes. If the database is non-operational, clear the database as described in "Clearing a database" on page 84.
Stopped	No nodes are running in the database.
Unknown	Cannot determine the state of the database.

Node Status

Use the `--nodes` option to make the `hadbm status` command display the following information for each node in the database:

- Node number
- Name of the machine where the node is running
- Port number of the node
- Role of the node. For a list of roles and their meanings, see "Roles of a Node" on page 94
- State of the node. For a list of states and their meanings, see "States of a Node" on page 94
- Number of the corresponding mirror node.

A node's role and state can change as described in these sections:

- "Roles of a Node" on page 94
- "States of a Node" on page 94

Roles of a Node

A node is assigned a role during its creation and can take any one of these roles:

- **Active:** Stores data and allows client access. Active nodes are in mirrored pairs.
- **Spare:** Allows client access, but does not store data. After initializing data devices, monitors other data nodes to initiate repair if another node becomes unavailable.
- **Offline:** Provide no services until their role changes. When placed back online, its role can change back to its former role.
- **Shutdown:** An intermediate step between active and offline, waiting for a spare node to take over its functioning. After the spare node has taken over, the node is taken offline.

States of a Node

A node can be in any one of the following states:

- **Starting:** The node is starting.
- **Waiting:** The node cannot decide its start level and is offline. If a single node is in this state for more than two minutes, stop the node and then start it at the `repair` level; see "Stopping a Node" on page 81 and "Starting a Node" on page 80 "Clearing a database" on page 84.
- **Running:** The node is providing all services that are appropriate for its role.
- **Stopping:** The node is in the process of stopping.
- **Stopped:** The node is inactive. Repair of a stopped node is prohibited.
- **Recovering:** The node is being recovered. When a node fails, the mirror node takes over the functions of the failed node. The failed node tries to recover by using the data and log records in main memory or on disk. The failed node uses the log records from the mirror node to catch up with the transactions performed when it was down. If recovery is successful, the node becomes active. If recovery fails, the node state changes to repairing.
- **Repairing:** The node is being repaired. This operation reinitializes the node and copies the data and log records from the mirror node. Repair is more time consuming than recovery.

Getting Device Information

Monitor free space in HADB data (disk storage) devices:

- Routinely, to check the trend in disk space use.
- As part of preventive maintenance: if the user load has increased and you want to resize or scale the database configuration.

- As part of scaling up the database: Before running `hadbm addnodes` to add new nodes to the system, check whether there is enough device space. Remember, you need around 40-50% free space on the existing nodes to add nodes.
- When you see messages in the history files and `server.log` file such as
  - `No free blocks on data devices`
  - `No unreserved blocks on data devices .`

Use the `hadbm deviceinfo` command to get information about free space in data devices. This command displays the following information for each node of the database:

- Total device size allocated, in MB (Totalsize).
- Free space in MB (Freesize).
- Percent of device currently being used (Usage)

The command syntax is:

```
hadbm deviceinfo  [--details]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]  [dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

The `--details` option displays the following additional information:

- Number of read operations by the device.
- Number of write operations by the device.
- Name of the device.

See "General Options" on page 61 for a description of other command options.

For more information, see `hadbm-deviceinfo`(1).

To determine the space available for user data, take the total device size, then subtract the space reserved for HADB: four times the `LogBufferSize` + `1%` of the device size. If you do not know the size of the log buffer, use the command `hadbm get logbufferSize`. For example, if the total device size is 128 MB and the `LogBufferSize` is 24 MB, the space available for user data is 128 – (4 x 24) = 32 MB. Of the 32 MB, half is used for replicated data and around one percent is used for the indices, and only 25 percent is available for the real user data.

The space available for user data is the difference between the total size and reserved size. If the data is refragmented in the future, the free size must be approximately equal to 50% of the space available for user data. If refragmentation is not relevant, the data devices can be exploited to their maximum. Resource consumption warnings are written to the history files when the system is running short on device space.

For more information about tuning HADB, see the *Sun Java System Application Server Performance Tuning Guide*.

EXAMPLE 3–16 Example of getting device information

The following command:

```
hadbm deviceinfo --details
```

Displays the following example results:

```
NodeNO Totalsize Freesize Usage NReads NWrites DeviceName
0      128       120      6%    10000  5000    C:\Sun\SUNWhadb\hadb.data.0
1      128       124      3%    10000  5000    C:\Sun\SUNWhadb\hadb.data.1
2      128       126      2%     9500  4500    C:\Sun\SUNWhadb\hadb.data.2
3      128       126      2%     9500  4500    C:\Sun\SUNWhadb\hadb.data.3
```

Getting Runtime Resource Information

The `hadbm resourceinfo` command displays HADB runtime resource information. You can use this information to help identify resource contention, and reduce performance bottlenecks. For details, see "Tuning HADB" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Performance Tuning Guide*.

The command syntax is:

```
hadbm resourceinfo  [--databuf]  [--locks]  [--logbuf]  [--nilogbuf]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
[dbname]
```

The *dbname* operand specifies the database name. The default is `hadb`.

The following table describes the `hadbm resourceinfo` special command options. See "General Options" on page 61 for a description of other command options.

For more information, see `hadbm-resourceinfo`(1).

TABLE 3–15 hadbm resourceinfo Command Options

Option	Description
--databuf	Display data buffer pool information.
-d	See "Data Buffer Pool Information" on page 97 below for more information.
--locks	Display lock information.
-l	See "Lock Information" on page 97 below for more information.

TABLE 3–15 hadbm resourceinfo Command Options *(Continued)*

Option	Description
--logbuf	Display log buffer information.
-b	See "Log Buffer Information" on page 98 below for more information.
--nilogbuf	Display node internal log buffer information.
-n	See "Node Internal Log Buffer Information" on page 98 below for more information.

Data Buffer Pool Information

Data buffer pool information contains the following:

- `NodeNo`: Node number.
- `Avail`: Total space available in the pool, in MBytes.
- `Free`: Free space available, in MBytes.
- `Access`: Cumulative number of accesses to the data buffer from database, from start until now.
- `Misses`: Cumulative number of page faults that have occurred from database start until now.
- `Copy-on-Write`: Cumulative number of pages copied internally in the data buffer due to checkpointing.

When a user transaction performs an operation on a record, the page containing the record must be in the data buffer pool. If it is not, a `miss` or a page fault occurs. The transaction then has to wait until the page is retrieved from the data device file on the disk.

If the miss rate is high, increase the data buffer pool. Since the misses are cumulative, run `hadbm resourceinfo` periodically and use the difference between two runs to see the trend of miss rate. Do not be concerned if free space is very small, since the checkpointing mechanism will make new blocks available.

EXAMPLE 3–17 Example data buffer pool information

For example:

```
NodeNO Avail Free Access Misses Copy-on-Write
0 256 128 100000 50000 10001 256 128 110000 45000 950
```

Lock Information

Lock information is as follows:

- `NodeNo`: Node Number.
- `Avail`: Total number of locks available on the node.
- `Free`: Number of free locks.

- `Waits`: Number of transactions waiting to acquire locks. This is cumulative.

One single transaction cannot use more than 25% of the available locks on a node. Therefore, transactions performing operations in large scale should be aware of this limitation. It is best to perform such transactions in batches, where each batch must be treated as a separate transaction, that is, each batch commits. This is needed because read operations running with `repeatable read` isolation level, and `delete`, `insert`, and `update` operations use locks that are released only after the transaction terminates.

To change the `NumberOfLocks`, see "Clearing and Archiving History Files" on page 101

EXAMPLE 3–18 Example lock information

For example:

```
NodeNO Avail Free Waits
0 50000 20000 101 50000 20000 0
```

Log Buffer Information

Log buffer information is:

- `NodeNo`: Node Number
- `Available`: amount of memory allocated for the log buffer in MB
- `Free`: amount of free memory in MB

Do not worry if free space is very small, since HADB starts compressing the log buffer. HADB starts compression from the head of the ring buffer and performs it on consecutive log records. Compression cannot proceed when HADB encounters a log record that has not been executed by the node and received by the mirror node

EXAMPLE 3–19 Example of log buffer information

For example:

```
NodeNO Avail Free
0 16 21 16 3
```

Node Internal Log Buffer Information

Node internal log buffer information is:

- Node Number
- Available: amount of memory allocated for the log device in MB
- Free: amount of free memory in MB

EXAMPLE 3–20 Example of internal log buffer information

For example:

```
NodeNO Avail Free

0 16 21 16 3
```

Maintaining HADB Machines

HADB achieves fault tolerance by replicating data on mirror nodes. In a production environment, a mirror node is on a separate DRU from the node it mirrors, as described in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide*.

A *failure* is an unexpected event such as a hardware failure, power failure, or operating system reboot. The HADB tolerates single failures: of one node, one machine (that has no mirror node pairs), one or more machines belonging to the same DRU, or even one entire DRU. However, HADB does *not* automatically recover from a double failure, which is the simultaneous failure of one or more mirror node pairs. If a double failure occurs, you must clear HADB and recreate its session store, which erases all its data.

There are different maintenance procedures, depending on whether you need to work on a single machine or multiple machines.

▼ To perform maintenance on a single machine

This procedure is applicable to both planned and unplanned maintenance, and does not interrupt HADB availability.

1. **Perform the maintenance procedure and get the machine up and running.**
2. **Ensure that `ma` is running.**

   If `ma` runs as a Windows service or under `init.d` scripts (recommended for deployment), it should have been started by the operating system. If not start it manually. See "Starting the Management Agent" on page 54.
3. **Start all nodes on the machine.**

   For more information, see "Starting a Node" on page 80.
4. **Check whether the nodes are active and running.**

   For more information, see "Getting the Status of HADB" on page 92

▼ To perform planned maintenance on all HADB machines

Planned maintenance includes operations such as hardware and software upgrades. This procedure does not interrupt HADB availability.

1. **For each spare machine in the first DRU, repeat the single machine procedure as described in "To perform maintenance on a single machine" on page 99, one by one, for each machine.**
2. **For each active machine in the first DRU, repeat the single machine procedure as described in "To perform maintenance on a single machine" on page 99, one by one, for each machine.**
3. **Repeat step 1 and step 2 for the second DRU.**

▼ To perform planned maintenance on all HADB machines

This procedure is applicable when HADB is on single or multiple machines. It interrupts HADB service during the maintenance procedure.

1. **Stop HADB. See "Stopping a Database" on page 83 .**
2. **Perform the maintenance procedure and get all the machines up and running.**
3. **Ensure `ma` is running.**
4. **Start HADB.**

   For more information, see "Starting a Database" on page 82.

   After you complete the last step, HADB data becomes available again.

▼ To perform unplanned maintenance in the event of a failure

- **Check the database state.**

  See "Getting the Status of HADB" on page 92

  - **If the database state is Operational or better:**

    The machines needing unplanned maintenance *do not* include mirror nodes. Follow the single machine procedure for each failed machine, one DRU at a time. HADB service is not interrupted.

  - **If the database state is Non-Operational:**

    The machines needing unplanned maintenance include mirror nodes. One such case is when the entire HADB is on a single failed machine. Get all the machines up and running first. Then clear HADB and recreate the session store. See "Clearing a database" on page 84. This interrupts HADB service.

Clearing and Archiving History Files

HADB history files record all database operations and error messages. HADB appends to the end of existing history files, so the files grow over time. To save disk space and prevent files from getting too large, periodically clear and archive history files.

To clear a database's history files, use the `hadbm clearhistory` command.

The command syntax is:

```
hadbm clearhistory
[--saveto=path]
[dbname]
[--adminpassword=password | --adminpasswordfile=file]
[--agent=maurl]
```

The *dbname* operand specifies the database name. The default is `hadb`.

Use the `--saveto` option (short form `-o`) to specify the directory in which to store the old history files. This directory must have appropriate write permissions. See "General Options" on page 61 for a description of other command options.

For more information, see `hadbm-clearhistory`(1).

The `--historypath` option of the `hadbm create` command determines the location of the history files. The names of the history files are of the format *dbname*`.out.` *nodeno*. For information on `hadbm create`, see "Creating a Database" on page 65

History File Format

Each message in the history file contains the following information:

- The abbreviated name of the HADB process that produced the message.
- The type of message:
  - `INF` - general information
  - `WRN` - warnings
  - `ERR` - errors
  - `DBG` - debug information
- A timestamp. The time is obtained from the host machine system clock.
- The service set changes occurring in the system when a node stops or starts.

Messages about resource shortages contain the string "HIGH LOAD."

You do not need a detailed knowledge of all entries in the history file. If for any reason you need to study a history file in greater detail, contact Sun customer support.

◆ ◆ ◆ CHAPTER 4

Configuring Load Balancing and Failover

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

How the Load Balancer Works

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

Using a load balancer also enables requests to fail over from one server instance to another. For HTTP session information to persist, configure HTTP session persistence. For more information, see Chapter 8, "Configuring High Availability Session Persistence and Failover"

For complete instructions on configuring load balancing, see the *Sun Java System Application Server High Availability Administration Guide*.

Use the asadmin tool, not the Admin Console, to configure HTTP load balancing.

See Also:

- "HTTP Load Balancing Algorithm" on page 104
- "Procedure to Set Up Load Balancing" on page 106

Assigned Requests and Unassigned Requests

When a request first comes in from an HTTP client to the load balancer, it is a request for a new session. A request for a new session is called an *unassigned* request. The load balancer routes this request to an application server instance in the cluster according to a round-robin algorithm.

Once a session is created on an application server instance, the load balancer routes all subsequent requests for this session only to that particular instance. A request for an existing session is called an *assigned* or a *sticky* request.

HTTP Load Balancing Algorithm

The Sun Java System Application Server load balancer uses a *sticky round robin algorithm* to load balance incoming HTTP and HTTPS requests. All requests for a given session are sent to the same application server instance. With a sticky load balancer, the session data is cached on a single application server rather than being distributed to all instances in a cluster.

Therefore, the sticky round robin scheme provides significant performance benefits that normally override the benefits of a more evenly distributed load obtained with a pure round robin scheme.

When a new HTTP request is sent to the load balancer plug-in, it's forwarded to an application server instance based on a simple round robin scheme. Subsequently, this request is "stuck" to this particular application server instance, either by using cookies, or explicit URL rewriting.

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

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

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

Sample Applications

The following directories contain sample applications that demonstrate load balancing and failover:

install_dir`/samples/ee-samples/highavailability`
install_dir`/samples/ee-samples/failover`

The `ee-samples` directory also contains information for setting up your environment to run the samples.

Setting Up HTTP Load Balancing

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

Prerequisites for Setting Up Load Balancing

Before configuring your load balancer, you must:

- Install a web server.
- Install the load balancer plug-in.

  For information on the installation procedure, see the *Sun Java System Application Server Installation Guide* (if using the stand-alone Application Server) or the *Sun Java Enterprise System Installation Guide* (if using Java Enterprise System).
- Configure the web server. For more information, see "Configuring Web Servers for Load Balancing" on page 107
- Create Application Server clusters or server instances to participate in load balancing.
- Deploy applications to these clusters or instances.

HTTP Load Balancer Deployments

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

Using Clustered Server Instances

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

If you have multiple clusters, requests are only load balanced and failed over between the instances in a single cluster. Use multiple clusters in a load balancer to easily enable rolling upgrades of applications. For more information, see "Upgrading Applications Without Loss of Availability" on page 125.

Using a Single, Stand-Alone Instance with Load Balancer Used as a Reverse-Proxy Plug-in

You can also configure your load balancer to use stand-alone server instance instead of a cluster. This configuration results in the load balancer plug-in working as a reverse-proxy plug-in (sometimes called a pass-through plug-in). When the web server receives requests for applications enabled in the load balancer, it forwards the requests directly to the Application Server.

Use the same procedures to configure the load balancer for a pass-through plug-in as you use to configure it for a cluster of server instances.

Using Multiple Stand-Alone Instances

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

Procedure to Set Up Load Balancing

Use the `asadmin` tool to configure load balancing in your environment. For more information on the `asadmin` commands used in these steps, see "Configuring the Load Balancer" on page 116

▼ To Set Up Load Balancing

1 **Create a load balancer configuration using the `asadmin` command `create-http-lb-config`.**

2. **Add a reference to a cluster or stand-alone server instance for the load balancer to manage using** `asadmin create-http-lb-ref`**.**

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

3. **Enable the cluster or stand-alone server instance referenced by the load balancer using** `asadmin enable-http-lb-server`**.**

4. **Enable applications for load balancing using** `asadmin enable-http-lb-application`**.**

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

5. **Create a health checker using** `asadmin create-health-checker`**.**

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

6. **Generate the load balancer configuration file using** `asadmin export-http-lb-config`**.**

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

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

Configuring Web Servers for Load Balancing

The load balancer plug-in installation program makes a few modifications to the web server's configuration files. The changes made depend upon the web server.

Note – The load balancer plug-in can be installed either along with Sun Java System Application Server Enterprise Edition, or separately, on a machine running the supported web server. For complete details on the installation procedure, see Chapter 1, "Installing Application Server Software," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Installation Guide* (if using the standalone Application Server) or Sun Java Enterprise System 2005Q5 Installation Guide (if using Java Enterprise System).

- "Modifications to Sun Java System Web Server" on page 108
- "Using Apache Web Server" on page 108
- "Installation" on page 41
- "Configuring Multiple Web Server Instances" on page 115

Modifications to Sun Java System Web Server

The installation program adds the following entries to the Sun Java System Web Server's configuration files:

To the web server instance's `magnus.conf` file, it adds:

```
##EE lb-pluginInit
fn="load-modules"
shlib="web_server_install_dir/plugins/lbplugin/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrough" Thread="no"
Init fn="init-passthrough"
##end addition for EE lb-plugin
```

To the web server instance's `obj.conf` file, it adds:

```
<Object name=default>
NameTrans fn="name-trans-passthrough" name="lbplugin"
config-file="web_server_install_dir/web_server_instance/config/loadbalancer.xml"
<Object name="lbplugin">
  ObjectType fn="force-type" type="magnus-internal/lbplugin"
  PathCheck fn="deny-existence" path="*/WEB-INF/*"
  Service type="magnus-internal/lbplugin"
  fn="service-passthrough"
  Error reason="Bad Gateway"
  fn="send-error"
  uri="$docroot/badgateway.html"
</object>
```

In the above code, `lbplugin` is a name that uniquely identifies the `Object`, and *web_server_install_dir/ web_server_instance/*`config/loadbalancer.xml` is the location of the XML configuration file for the virtual server on which the load balancer is configured to run.

After installing, configure the load balancer as described in "Setting Up HTTP Load Balancing" on page 105

Using Apache Web Server

To use Apache Web Server, you must perform certain configuration steps before installing the load balancer plug-in. The load balancer plug-in installation also makes additional modifications to the Apache Web Server. After the plug-in is installed, you must perform additional configuration steps.

Note – On Apache 1.3, when more than one Apache child processes runs, each process has its own load balancing round robin sequence. For example, if there are two Apache child processes running, and the load balancer plug-in load balances on to two application server instances, the first request is sent to instance #1 and the second request is also sent to instance #1. The third request is sent to instance #2 and the fourth request is sent to instance #2 again. This pattern is repeated (instance1, instance1, instance2, instance2, etc.)This behavior is different from what you might expect, that is, instance1, instance2, instance1, instance2, etc. In Sun Java System Application Server, the load balancer plug-in for Apache instantiates a load balancer instance for each Apache process, creating an independent load balancing sequence.

Apache 2.0 has multithreaded behavior if compiled with the `--with-mpm=worker` option.

Requirements for Using Apache Web Server

For the Apache Web Server, your installation must meet the minimum requirements, depending on the version of Apache.

Requirements for Apache 1.3

With Apache 1.3, the load balancer plug-in requires:

- `openssl-0.9.7e` (source)
- `mod_ssl-2.8.16-1.3.`*x* (source), where *x* represents the version of Apache. The mod_ssl version must match the Apache version.
- `gcc-3.3-sol9-sparc-local` packages (for Solaris SPARC)
- `gcc-3.3-sol9-intel-local` packages (for Solaris x86)
- `flex-2.5.4a-sol9-sparc-local` packages (for Solaris SPARC)
- `flex-2.5.4a-sol9-intel-local` packages (for Solaris x86)

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

  In addition, before compiling Apache:
- On the Linux platform, install Sun Java System Application Server on the same machine.
- On the Solaris operating system, ensure that `gcc` version 3.3 and `make` are in the `PATH`, and `flex` is installed.
- On the Solaris 10 operating system, before running `make` for OpenSSL, run `mkheaders`, located in `/usr/local/lib/gcc-lib/sparc-sun-solaris2.9/3.3/install-tools` on Solaris SPARC or `/usr/local/lib/gcc-lib/i386-pc-solaris2.9/3.3/install-tools` on Solaris x86.

- If you are using gcc on Red Hat Enterprise Linux Advanced Server 2.1, the version must be later than gcc 3.0.

Note – To use C compiler other than gcc, set the path of the C compiler and make utility in the PATH environment variable. For example, with the sh shell: `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:appserver_installdir/lib`

Minimum Requirements for Apache 2

With Apache 2.0, the load balancer plug-in requires:

- `openssl-0.9.7e (source)`
- `httpd-2.0.49 (source)`
- `gcc-3.3-sol9-sparc-local` packages (for Solaris SPARC).
- `gcc-3.3-sol9-intel-local` packages (for Solaris x86)
- `flex-2.5.4a-sol9-sparc-local` packages (for Solaris SPARC)
- `flex-2.5.4a-sol9-intel-local` packages (for Solaris x86)

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

In addition, before compiling Apache:

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

Note – To use a C compiler other than gcc, set the path of the C compiler and make utility in the PATH environment variable. For example, with the sh shell:`export LD_LIBRARY_PATH=`*app_server_install_dir*`/lib:$LD_LIBRARY_PATH`.

Configuration before Installing the Load Balancer Plug-in

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

▼ To Install SSL-aware Apache

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

1. **Download and unpack the OpenSSL source.**

2. **Compile and build OpenSSL.**

   This step is not required on the Linux platform if OpenSSL 0.9.7.e is installed.

   Enter these commands:

   ```
   cd openssl-0.9.7e
   make
   make install
   ```

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

3. **Follow one of these procedures, depending on the version of Apache:**

   - **For Apache 1.3, configure Apache with `mod_ssl` with the following steps:**

     a. **Unpack the `mod_ssl` source.**

     b. **`cd mod_ssl-2.8.14–1.3.`*x***

     c. **`./configure –with-apache=../apache_1.3.`*x* `--with-ssl=../openssl-0.9.7e --prefix=`*install_path* `--enable-module=ssl --enable-shared=ssl --enable-rule=SHARED_CORE --enable-module=so`**

     In the above commands, *x* is the Apache version number, and *install_path* is the directory in which to install Apache.

     For more information on `mod_ssl`, see `http://www.modssl.org`.

   - **For Apache 2.0, configure the source tree:**

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

     b. **Run** `./configure --with-ssl=` *open_ssl_install_path* `--prefix=` *install_path* `--enable-ssl --enable-so`

     In the above commands, *x* is the Apache version number, *open_ssl_install_path* is the directory where OpenSSL is installed, and *install_path* is the directory in which to install Apache.

4. **For Apache on Linux 2.1, before compiling:**

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

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

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

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

For example:

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

c. Set environment variable LD_LIBRARY_PATH.

With all installations, set it to: *appserver_install_dir*/**lib**

With Java Enterprise System Installations, set it to *appserver_install_dir*/**lib:opt/sun/private/lib** .

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

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

In general the steps are:

a. `make`

b. `make certificate` **(Apache 1.3 only)**

c. `make install`

The command `make certificate` asks for a secure password. Remember this password as it is required for starting secure Apache.

6 Configure Apache for your environment.

Modifications Made by the Application Server Installer

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

- For Apache 1.3, the directory is `libexec`.

- For Apache 2.0, the directory is `modules` .

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

```
<VirtualHost machine_name:443>
##Addition for EE lb-plugin
LoadFile /usr/lib/libCstd.so.1
LoadModule apachelbplugin_module libexec/mod_loadbalancer.so
#AddModule mod_apachelbplugin.cpp
<IfModule mod_apachelbplugin.cpp>
  config-file webserver_instance/conf/loadbalancer.xml
  locale en
</IfModule>
<VirtualHost machine_ip_address>
  DocumentRoot "webserver_instance/htdocs"
  ServerName server_name
</VirtualHost>
##END EE LB Plugin ParametersVersion 7
```

▼ To configure Apache Security Files to work with the Load Balancer

Apache Web Server must have the correct security files to work well with the load balancer plug-in.

1. **Create a directory called** `sec_db_files` **under** *apache_install_dir***.**
2. **Copy** *application_server_domain_dir* `/config/*` **.db to** *apache_install_dir* `/sec_db_files` **.**
3. **Depending on the platform, perform additional configuration.**

   - **On the Solaris platform:**

     Add the path `/usr/lib/mps/secv1` to LD_LIBRARY_PATH in the *apache_install_dir*`/bin/apachectl` script. The path must be added before `/usr/lib/mps`.

   - **On Linux:**

     Add the path `/opt/sun/private/lib` to LD_LIBRARY_PATH in the *apache_install_dir*`/bin/apachectl` script. The path must be added before `/usr/lib`.

   - **On Microsoft Windows:**

     a. **Add a new path to the Path environment variable.**

        Click Start->Settings->Control Panel->System->Advanced->Environment Variables->System Variables.

        Add *application_server_install_dir* `/bin` to the Path environment variable.

b. Set the environment variable NSPR_NATIVE_THREADS_ONLY to 1.

In the Environment Variables window, under System Variables, click New. Enter Variable name of NSPR_NATIVE_THREADS_ONLY and Variable value of 1.

c. Restart the machine.

Modifications to Microsoft IIS

To configure Microsoft Internet Information Services (IIS) to use the load balancer plug-in, modify certain properties in Windows Internet Services Manager. The Internet Services Manager is located in the Administrative Tools folder in the Control Panel folder.

Make these modifications after installing the Sun Java System Application Server.

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

1 Open the Internet Services Manager.

2 Select the web site for which you want to enable the plug-in.

This web site is typically named the Default Web Site.

3 Right click on the web site and select Properties to open the Properties notebook.

4 Add a new ISAPI filter, following these steps:

a. Open the ISAPI Filters tab.

b. Click Add.

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

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

e. Click OK, and close the Properties notebook.

5 Create and configure a new virtual directory:

a. Right click on the default web site, select New, and then Virtual Directory.

The Virtual Directory Creation Wizard opens.

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

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

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

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

e. **Click Finish.**

6 **Add the path of `sun-passthrough.dll` file and *application_server_install_dir* `/bin` to the system's `PATH` environment variable.**

7 **Restart the machine.**

8 **Stop and start the web server for the new settings to take effect.**

To stop the web server, right click on the web site and select `Stop` . To start the web server, right click on the web site and select `Start`.

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

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

Automatically configured Sun-passthrough properties

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

Property	Definition	Default Value
lb-config-file	Path to the load balancer configuration file	*IIS_www_root*`\sun-passthrough\loadbalar`
log-file	Path to the load balancer log file	*IIS_www_root*`\sun-passthrough\lb.log`
log-level	Log level for the web server	INFO

Configuring Multiple Web Server Instances

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

▼ To Configure Multiple Web Server Instances

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

   Follow the steps in "Modifications to Sun Java System Web Server" on page 108 or "Using Apache Web Server" on page 108 , or "Installation" on page 41

2. **Copy the DTD file.**

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

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

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

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

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

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

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

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

     For information on creating a load balancer configuration and exporting it to a `loadbalancer.xml` file, see "Creating an HTTP Load Balancer Configuration" on page 117

Configuring the Load Balancer

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

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

This section describes how to create, modify, and use a load balancer configuration, including the following topics:

Creating an HTTP Load Balancer Configuration

Create a load balancer configuration using the `asadmin` command `create-http-lb-config`. "Creating an HTTP Load Balancer Configuration" on page 117 describes the parameters. For more information see the documentation for `create-http-lb-config`, `delete-http-lb-config`, and `list-http-lb-configs`.

TABLE 4–1 Load Balancer Configuration Parameters

Parameter	Description
response timeout	Time in seconds within which a server instance must return a response. If no response is received within the time period, the server is considered unhealthy. The default is 60.
HTTPS routing	Whether HTTPS requests to the load balancer result in HTTPS or HTTP requests to the server instance. For more information, see "Configuring HTTPS Routing" on page 123.
reload interval	Interval between checks for changes to the load balancer configuration file `loadbalancer.xml`. When the check detects changes, the configuration file is reloaded. A value of 0 disables reloading.For more information, see "Enabling Dynamic Reconfiguration" on page 121
monitor	Whether monitoring is enabled for the load balancer.
routecookie	Name of the cookie the load balancer plug-in uses to record the route information. The HTTP client must support cookies. If your browser is set to ask before storing a cookie, the name of the cookie is JROUTE.
target	Target for the load balancer configuration. If you specify a target, it is the same as adding a reference to it. Targets can be clusters or stand-alone instances.

Creating an HTTP Load Balancer Reference

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

Create a reference using `create-http-lb-ref`. You must supply the load balancer configuration name and the target server instance or cluster.

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

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

Enabling Server Instances for Load Balancing

After creating a reference to the server instance or cluster, enable the server instance or cluster using `enable-http-lb-server` . If you used a server instance or cluster as the target when you created the load balancer configuration, you must enable it.

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

Enabling Applications for Load Balancing

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

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

For more information, see the online help for `enable-http-lb-application` .

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

Creating the HTTP Health Checker

The load balancer's health checker periodically checks all the configured Application Server instances that are marked as unhealthy. A health checker is not required, but if no health checker exists, or if the health checker is disabled, the periodic health check of unhealthy instances is not performed.

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

Creating a Health Checker

To create the health checker, use the `asadmin create-http-health-checke r` command. Specify the following parameters:

TABLE 4–2 Health Checker Parameters

Parameter	Description	Default
url	Specifies the listener's URL that the load balancer checks to determine its state of health.	"/"
interval	Specifies the interval in seconds at which health checks of instances occur. Specifying 0 disables the health checker.	30 seconds
timeout	Specifies the timeout interval in seconds within which a response must be obtained for a listener to be considered healthy.	10 seconds

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

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

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

Additional Health Check Properties for Healthy Instances

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

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

To check healthy instances, set the following properties:

TABLE 4–3 Health-checker Manual Properties

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

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

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

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

Exporting the Load Balancer Configuration File

The load balancer plug-in shipped with Sun Java System Application Server uses a configuration file called `loadbalancer.xml`. Use the `asadmin` tool to create a load balancer configuration in the `domain.xml` file. After configuring the load balancing environment, export it to a file.

▼ To export the load balancer configuration

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

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

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

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

For example, for the Sun Java System Web Server, that location might be *web_server_root*`/config` .

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

Changing the Load Balancer Configuration

If you change a load balancer configuration by creating or deleting references to servers, deploying new applications, enabling or disabling servers or applications, and so on, export the load balancer configuration file again and copy it to the web server's `config` directory. For more information, see "Exporting the Load Balancer Configuration File" on page 120

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

Enabling Dynamic Reconfiguration

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

To enable dynamic reconfiguration:

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

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

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

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

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

The error log for Sun Java System Web Server' is at: *web_server_install_dir*`/`*webserver_instance*`/logs/`.

Disabling (Quiescing) a Server Instance or Cluster

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

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

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

▼ To disable a server instance or cluster

1 **Run** `asadmin disable-http-lb-server`**, setting the timeout (in minutes).**

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

3 **Copy the exported configuration to the web server** `config` **directory.**

4 **Stop the server instance or instances.**

Disabling (Quiescing) an Application

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

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

  When you disable an application from every server instance or cluster the load balancer references, the users of the disabled application experience loss of service until the application is enabled again. If you disable the application from one server instance or cluster while keeping it enabled in another server instance or cluster, users can still access the application.

▼ To disable an application

1 **Use** `asadmin disable-http-lb-application`**, specifying the following:**

- Timeout (in minutes).
- Name of the application to disable.

- Target cluster or instance on which to disable it.

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

3 **Copy the exported configuration to the web server** `config` **directory.**

Configuring HTTP and HTTPS Failover

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

This section discusses the following topics:

- "HTTPS Routing" on page 123
- "Configuring Idempotent URLs" on page 124

HTTPS Routing

The HTTP Secure (HTTPS) protocol uses Secure Sockets Layer (SSL) to provide encryption an decryption of HTTP requests for secure communication. For HTTPS routing to work, one or more HTTPS listeners must be configured.

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

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

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

Configuring HTTPS Routing

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

Note – If `https-routing` is set to `true`, and a new or a sticky request comes in where there are no healthy HTTPS listeners in the cluster, then that request generates an error.

Known Issues

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

- If a session uses a combination of HTTP and HTTPS requests, then the first request must be an HTTP Request. If the first request is an HTTPS request, it cannot be followed by an HTTP request. This is because the cookie associated with the HTTPS session is not returned by the browser. The browser interprets the two different protocols as two different servers, and initiates a new session.

  This limitation is valid only if `httpsrouting` is set to `true` .
- If a session has a combination of HTTP and HTTPS requests, then the application server instance must be configured with both HTTP and HTTPS listeners.

  This limitation is valid only if `httpsrouting` is set to `true`.
- If a session has a combination of HTTP and HTTPS requests, then the application server instance must be configured with HTTP and HTTPS listeners that use standard port numbers, that is, 80 for HTTP, and 443 for HTTPS. This limitation applies regardless of the value set for `httpsrouting`.

Configuring Idempotent URLs

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

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

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

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

For more information on configuring idempotent URLs, see "Configuring Idempotent URL Requests" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Developer's Guide*.

Upgrading Applications Without Loss of Availability

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

Application Compatibility

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

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

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

You can perform a rolling upgrade of a compatible application in either a single cluster or multiple clusters. For more information, see "Upgrading In a Single Cluster" on page 125

If the two versions of an application do not meet all the above criteria, then the applications are considered *incompatible.* Executing incompatible versions of an application in one cluster can corrupt application data and cause session failover to not function correctly. The problems depend on the type and extent of the incompatibility. It is good practice to upgrade an incompatible application by creating a "shadow cluster" to which to deploy the new version and slowly quiesce the old cluster and application. For more information, see "Upgrading Incompatible Applications" on page 129

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

Upgrading In a Single Cluster

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

▼ To upgrade an application in a single cluster

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

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

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

To do this with Admin Console:

a. **Expand the Configurations node.**

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

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

d. **Click Save**

Alternatively, use this command:

`asadmin set --user` *user* **`--passwordfile`** *password_file cluster_name* **`-config.dynamic-reconfiguration-enabled=false`**

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

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

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

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

Follow these steps:

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

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

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

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

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

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

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

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

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

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

Follow these steps:

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

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

c. **Copy the configuration file to the web server's configuration directory as described in** **"Upgrading In a Single Cluster" on page 125 of "Upgrading In a Single Cluster" on page 125 .**

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

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

Upgrading in Multiple Clusters

▼ To upgrade a compatible application in two or more clusters:

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

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

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

To do this with Admin Console:

a. **Expand the Configurations node.**

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

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

d. Click Save

e. Repeat for the other clusters

Alternatively, use this command:

`asadmin set --user` *user* `--passwordfile` *password_file* *cluster_name*`-config.dynamic-reconfiguration-enabled=false`

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

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

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

5 Quiesce one cluster from the load balancer

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

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

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

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

d. Wait until the timeout has expired.

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

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

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

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

8 Re-enable the cluster in load balancer:

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

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

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

9 Repeat steps 5 through 8 for the other clusters.

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

Upgrading Incompatible Applications

For information on what makes applications compatible, see "Application Compatibility" on page 125 the new version of the application is incompatible with the old. Also, you must upgrade incompatible application in two or more clusters. If you have only one cluster, create a "shadow cluster" for the upgrade, as described below.

When upgrading an incompatible application:

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

▼ To upgrade an incompatible application by creating a second cluster

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

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

2 Create a "shadow cluster" on the same or a different set of machines as the existing cluster.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Monitoring the HTTP Load Balancer Plug-in

Configuring Log Messages

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

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

Types of Log Messages

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

Load Balancer Configurator Log Messages

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

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

- When the log level is set to FINE:

  ```
  CONFxxxx: IdempotentUrlPattern configured <url-pattern> <no-of-retries> for
  web-module : <web-module>
  ```

- When the log level is set to SEVERE:

  ```
  CONFxxxx: Duplicate entry of Idempotent URL element <url-pattern> for
  webModule <web-module> in loadbalancer.xml."
  ```

- When the log level is set to WARN:

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

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

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

Request Dispatch and Runtime Log Messages

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

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

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

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

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

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

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

Configurator Error Messages

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

- Log level set to INFO:

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

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

- Log level set to FINE:

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

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

Enabling Load Balancer Logging

The load balancer plug-in logs the following information:

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

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

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

▼ To turn on load balancer logging

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

   - **With Sun Java System Web Server**

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

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

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

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

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

Understanding Monitoring Messages

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

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

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

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

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

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

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

  For example:

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

Note – In the RequestExit message, <response-time> indicates the total request turn-around time in milliseconds, from the perspective of the load balancer plug-in.

- The list of unhealthy instances, as follows:

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

  For example:

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

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

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

  For example:

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

◆ ◆ ◆ CHAPTER 5

Using Application Server Clusters

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

Overview of Clusters

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

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

Working with Clusters

- "To Upgrade Components Without Loss of Service" on page 142

To Create a Cluster

1. **In the tree component, select the Clusters node.**
2. **On the Clusters page, click New.**
   The Create Cluster page appears.
3. **In the Name field, type a name for the cluster.**
   The name must:
   - Consist only of uppercase and lowercase letters, numbers, underscores, hyphens, and periods (.)
   - Be unique across all node agent names, server instance names, cluster names, and configuration names
   - Not be `domain`
4. **In the Configuration field, choose a configuration from the drop-down list.**
   - **To create a cluster that does not use a shared configuration, choose** `default-config`**.**
     Leave the radio button labeled "Make a copy of the selected Configuration" selected. The copy of the default configuration will have the name *cluster_name*`-config`.
   - **To create a cluster that uses a shared configuration, choose the configuration from the drop-down list.**
     Select the radio button labeled "Reference the selected Configuration" to create a cluster that uses the specified existing shared configuration.
5. **Optionally, add server instances.**
   You can also add server instances after the cluster is created.

   Before you create server instances for the cluster, first create one or more node agents or node agent placeholders. See "To Create a Node Agent Placeholder" on page 165

   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 138
- "To Create Server Instances for a Cluster" on page 137
- "To Configure Applications for a Cluster" on page 140
- "To Configure Resources for a Cluster" on page 140
- "To Delete a Cluster" on page 141
- "To Upgrade Components Without Loss of Service" on page 142

For more details on how to administer clusters, server instances, and node agents, see "Deploying Node Agents" on page 155.

▼ 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 165

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 153
- "To Create a Cluster" on page 136

- "To Configure a Cluster" on page 138
- "To Configure Applications for a Cluster" on page 140
- "To Configure Resources for a Cluster" on page 140
- "To Delete a Cluster" on page 141
- "To Upgrade Components Without Loss of Service" on page 142
- "To Configure Server Instances in a Cluster" on page 139

▼ 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 136
- "To Create Server Instances for a Cluster" on page 137
- "To Configure Applications for a Cluster" on page 140
- "To Configure Resources for a Cluster" on page 140
- "To Delete a Cluster" on page 141
- "To Upgrade Components Without Loss of Service" on page 142
- "To Migrate EJB Timers" on page 141

▼ 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 136
- "To Configure a Cluster" on page 138
- "To Create Server Instances for a Cluster" on page 137
- "To Configure Applications for a Cluster" on page 140
- "To Configure Resources for a Cluster" on page 140
- "To Delete a Cluster" on page 141
- "To Upgrade Components Without Loss of Service" on page 142
- "Recovering Transactions" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 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 136
- "To Configure a Cluster" on page 138
- "To Create Server Instances for a Cluster" on page 137
- "To Configure Resources for a Cluster" on page 140
- "To Delete a Cluster" on page 141
- "To Upgrade Components Without Loss of Service" on page 142

▼ To Configure Resources for a Cluster

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

2 **Select the node for the cluster.**

3 **Click the Resources tab to bring up the Resources page.**

On this page, you can:

- Create a new resource for the cluster: from the New drop-down list, select a type of resource to create. Make sure to specify the cluster as a target when you create the resource.
- Enable or Disable a resource globally: select the checkbox next to a resource and click Enable or Disalbe. This action does not remove the resource.
- Display only resources of a particular type: from the Filter drop-down list, select a type of resource to display in the list.
- Edit a resource: click the resource name.

See Also
- "To Create a Cluster" on page 136
- "To Configure a Cluster" on page 138
- "To Create Server Instances for a Cluster" on page 137
- "To Configure Applications for a Cluster" on page 140
- "To Delete a Cluster" on page 141

▼ 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 136
- "To Configure a Cluster" on page 138
- "To Create Server Instances for a Cluster" on page 137
- "To Configure Applications for a Cluster" on page 140
- "To Configure Resources for a Cluster" on page 140
- "To Upgrade Components Without Loss of Service" on page 142

▼ 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 138
- "Configuring the EJB Timer Service Settings" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*

▼ To Upgrade Components Without Loss of Service

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

This approach is not possible if:

- You change the schema of the high-availability database (HADB). For more information, see Chapter 3, "Administering High Availability Database"
- You perform an application upgrade that involves a change to the application database schema.

Caution – Upgrade all server instances in a cluster together. Otherwise, there is a risk of version mismatch caused by a session failing over from one instance to another where the instances have different versions of components running.

1 **Stop one of the clusters using the Stop Cluster button on the General Information page for the cluster.**

2 **Upgrade the component in that cluster.**

3 **Start the cluster using the Start Cluster button on the General Information page for the cluster.**

4 **Repeat the process with the other clusters, one by one.**

Because sessions in one cluster will never fail over to sessions in another cluster, there is no risk of version mismatch caused by a session's failing over from a server instance that is running one version of the component to another server instance (in a different cluster) that is running a different version of the component. A cluster in this way acts as a safe boundary for session failover for the server instances within it.

See Also

- "To Create a Cluster" on page 136
- "To Configure a Cluster" on page 138
- "To Create Server Instances for a Cluster" on page 137
- "To Configure Applications for a Cluster" on page 140
- "To Configure Resources for a Cluster" on page 140
- "To Delete a Cluster" on page 141

◆ ◆ ◆ CHAPTER 6

Managing Named Configurations

This chapter describes adding, changing, and using named server configurations in Application Server. It contains the following sections:

About Named Configurations

Named Configurations

A named configuration is a set of server configuration information, including settings for things such as HTTP listeners, ORB/IIOP listeners, JMS brokers, the EJB container, security, logging, and monitoring. Applications and resources are not defined in named configurations.

Configurations are created in the administration 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 named configuration contains so many required configuration 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 146
- "Configurations Created when Creating Instances or Clusters" on page 146
- "Unique Port Numbers and Configurations" on page 147
- "To Create a Named Configuration" on page 148
- "Editing a Named Configuration's Properties" on page 148

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. No unclustered server instances or clusters are allowed to refer to the `default-config` configuration; 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 146
- "Named Configurations" on page 145
- "To Create a Named Configuration" on page 148
- "Editing a Named Configuration's Properties" on page 148
- "To Edit Port Numbers for Instances Referencing a Configuration" on page 150

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 146
- "Named Configurations" on page 145
- "To Create a Named Configuration" on page 148
- "Editing a Named Configuration's Properties" on page 148

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 150
- "Editing a Named Configuration's Properties" on page 148
- "Named Configurations" on page 145

Working with Named Configurations

▼ 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

- "Named Configurations" on page 145
- "The default-config Configuration" on page 146
- "Editing a Named Configuration's Properties" on page 148
- "To Edit Port Numbers for Instances Referencing a Configuration" on page 150
- "To view a Named Configuration's Targets" on page 150
- "To Delete a Named Configuration" on page 151

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. **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.
4. **Add, delete, or modify properties as desired.**
5. **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

- "Named Configurations" on page 145
- "To Create a Named Configuration" on page 148
- "To view a Named Configuration's Targets" on page 150
- "To Delete a Named Configuration" on page 151

▼ 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.**

   The Admin Console displays the Configuration System Properties page.
3. **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.
4. **Change the values as desired and click Save.**

More Information

Equivalent asadmin command

```
set
```

See Also

- "Unique Port Numbers and Configurations" on page 147
- "Named Configurations" on page 145
- "Editing a Named Configuration's Properties" on page 148

▼ 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.**

See Also

- "Unique Port Numbers and Configurations" on page 147
- "Named Configurations" on page 145
- "To Create a Named Configuration" on page 148
- "Editing a Named Configuration's Properties" on page 148

- "To Delete a Named Configuration" on page 151

▼ 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

- "Named Configurations" on page 145
- "To Create a Named Configuration" on page 148
- "Editing a Named Configuration's Properties" on page 148
- "To view a Named Configuration's Targets" on page 150

◆ ◆ ◆ CHAPTER 7

Configuring Node Agents

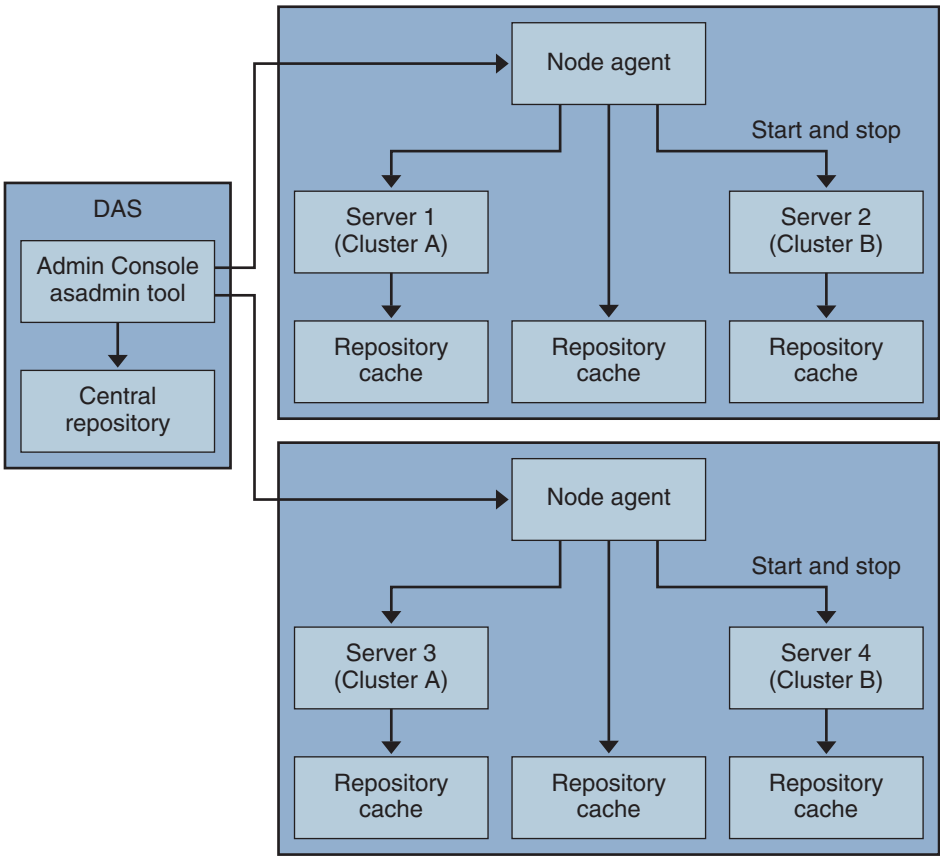
This chapter describes the node agents in Application Server. It contains the following sections:

What Is a Node Agent?

A node agent is a lightweight process that is required on every machine that hosts server instances, including the machine that hosts the Domain Administration Server (DAS). The node agent:

- Starts, stops, creates and deletes server instances as instructed by the Domain Administration Server.
- Restarts failed server instances.
- Provides a view of the log files of failed servers.
- Synchronizes each server instance's local configuration repository with the Domain Administration Server's central repository. Each local repository contains only the information pertinent to that server instance or node agent.

  The following figure illustrates the overall node agent architecture:



When you install the Application Server, a node agent is created by default with the host name of the machine. This node agent must be manually started on the local machine before it runs.

You can create and delete server instances even if the node agent is not running. However, the node agent must be running before you use it to start and stop server instances.

If you stop the node agent, the server instances it manages are stopped too.

A node agent services a single domain. If a machine hosts instances running in multiple domains, it must run multiple node agents.

See Also

- "Deploying Node Agents" on page 155
- "Node Agent Placeholders" on page 155
- "Node Agent and Domain Administration Server Synchronization" on page 158
- "To Create a Node Agent Placeholder" on page 165

- "Creating a Node Agent" on page 169
- "Starting a Node Agent" on page 170
- "Stopping a Node Agent" on page 171
- "Deleting a Node Agent" on page 171

Node Agent Placeholders

You can create and delete server instances without an existing node agent using a node agent placeholder. The placeholder is a node agent configuration created on the Domain Administration Server (DAS) before the node agent itself is created on the node agent's local system.

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 169 and "Starting a Node Agent" on page 170

See Also:

- "To Create a Node Agent Placeholder" on page 165
- "What Is a Node Agent?" on page 153
- "Deploying Node Agents" on page 155
- "Node Agent and Domain Administration Server Synchronization" on page 158

Deploying Node Agents

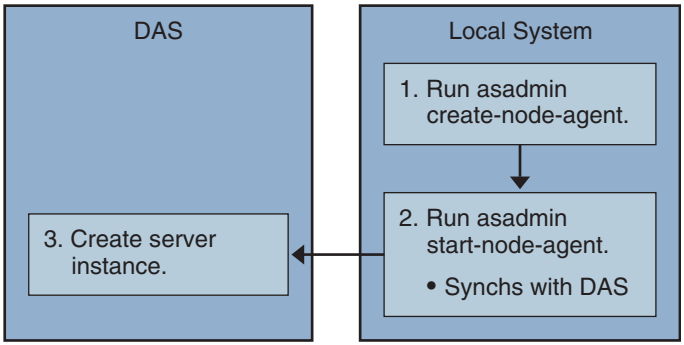
You configure and deploy node agents in two ways:

- *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:



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 169 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.

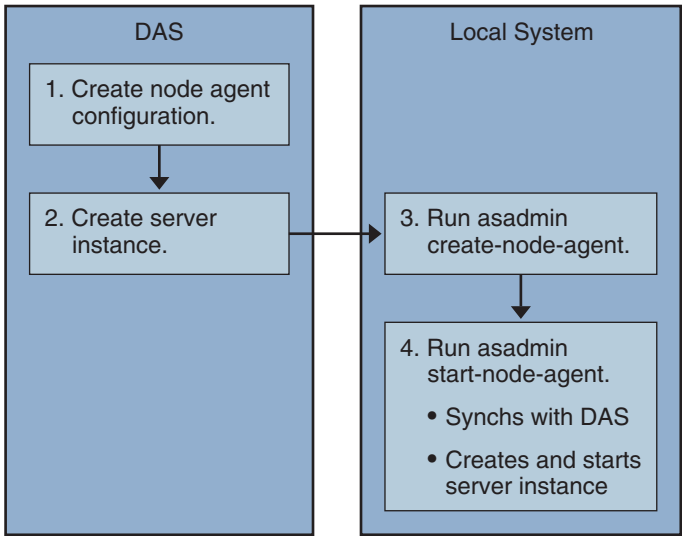
See "Starting a Node Agent" on page 170 for more information.

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.



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 165 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 169 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 170 for more information.

See Also

- "What Is a Node Agent?" on page 153
- "Node Agent Placeholders" on page 155
- "Node Agent and Domain Administration Server Synchronization" on page 158

Node Agent and Domain Administration Server Synchronization

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

When a node agent is started for the first time, it sends a request to the Domain Administration Server (DAS) for the latest information in the central repository. Once it successfully contacts the DAS and gets configuration information, the node agent is bound to that DAS.

Note – By default, the `asadmin start-node-agent` command automatically starts the remote server instances without synchronizing with DAS. If you are starting a remote server instance that is synchronized with the central repository managed by DAS, specify the `--startinstances=false` option of the `asadmin start-node-agent` command. Then use the `asadmin start-instance` command to start the remote server instance.

If you created a placeholder node agent on the DAS, when the node agent is started for the first time it gets its configuration from the central repository of the DAS. During its initial start-up, if the node agent is unable to reach the DAS because the DAS is not running, the node agent stops and remains unbound.

If changes are made in the domain to the node agent's configuration, they are automatically communicated to the node agent on the local machine while the node agent is running.

If you delete a node agent configuration on the DAS, the next time the node agent synchronizes, it stops and marks itself as awaiting deletion. Manually delete it using the local `asadmin delete-node-agent` command.

Server Instance Synchronization

If you explicitly start a server instance with the Admin Console or `asadmin` tool, the server instance is synchronized with the central repository. If this synchronization fails, the server instance doesn't start.

If a node agent starts a server instance without an explicit request through the Admin Console or the `asadmin` tool, the repository cache for the server instance is not synchronized. The server instance runs with the configuration as stored in its cache. You must not add or remove files in the remote server instance's cache.

The remote server instance's configuration are treated as cache (all files under `nodeagents`/*na1*/*server1*) and owned by Application Server. In extreme cases, if user removes all files of a remote server instance and restarts the node agent, the remote server instance (for example, *server1*) will be recreated and all necessary files will be synchronized.

The following files and directories are kept synchronized by the Application Server.

TABLE 7–1 Files and directories synchronized among remote server instances

File or directory	Description
`applications`	All deployed applications. The parts of this directory (and sub directories) synchronized depend on the applications referred to from the server instance. The Node agent does not synchronize any of the applications because it does not reference any application.
`config`	Contains configuration files for the entire domain. All the files in this directory are synchronized except runtime temporary files, such as, `admch`, `admsn`, `secure.seed`, `.timestamp`, and `__timer_service_shutdown__.dat`.
`config`/*config_name*	Directory to store files to be shared by all instances using config named *config_name*. There will be one such directory for every config defined in domain.xml. All the files in this directory are synchronized to the server instances that are using the *config_name*.
`config`/*config_name*`/lib/ext`	Folder where Java extension classes (as zip or jar archives) can be dropped. This is used by applications deployed to server instances using config named *config_name*. These jar files are loaded using Java extension mechanism.
docroot	The HTTP document root. In out of the box configuration, all server instances in the domain use the same docroot. The docroot property of the virtual server needs to be configured to make the server instances use a different docroot.

TABLE 7–1 Files and directories synchronized among remote server instances *(Continued)*

File or directory	Description
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.

Synchronizing Library Files

The `--libraries` deploy time attribute for an application can be used to specify runtime dependencies of an application.

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 Classloader" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 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/config/cluster1`, `config/lib/ext` - adds to java.ext.dirs (`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) that are synchronized by the Application Server, use the `doNotRemoveList` flag. This attribute takes a coma-separated list of files or directories. Your application dependent files are not removed during server startup, even if they do not exist in the central repository managed by DAS. If the same file exists in the central repository, they will be over written during synchronization.

Use the `INSTANCE-SYNC-JVM-OPTIONS` property to pass in the doNotRemoveList attribute.

For example:

```
<node-agent name="na1" ...>

...

<property name="INSTANCE-SYNC-JVM-OPTIONS"
value="-Dcom.sun.appserv.doNotRemoveList=applications/j2ee-modules
/<webapp_context>/logs,generated/mylogdir"/>

</node–agent>
```

Viewing Node Agent Logs

Each node agent has its own log file. If you experience problems with a node agent, see the log file at:

node_agent_dir /*node_agent_name*/`agent/logs/server.log` .

Sometimes the node agent log instructs you to look at a server's log to get a detailed message about a problem.

The server logs are located at:

node_agent_dir/*node_agent_name*/ *server_name*/`logs/server.log`

The default location for *node_agent_dir* is *install_dir*/`nodeagents`.

Tasks Available through the Admin Console and asadmin Tool

For node agents, some tasks must be performed locally on the system where the node agent runs, while others can be performed on the Domain Administration Server. Tasks that need to be performed locally are only available through the `asadmin` tool running on the machine where the node agent resides. Tasks that operate on the Domain Administration Server are available through the Admin Console and through the `asadmin` tool

The following table summarizes the tasks and where to run them:

TABLE 7–2 Tasks available through the Admin Console and the asadmin command

Task	Admin Console	asadmin Command
Create node agent placeholder/configuration 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

For more information, see:

- "What Is a Node Agent?" on page 153
- "Deploying Node Agents" on page 155
- "To Create a Node Agent Placeholder" on page 165
- "To Delete a Node Agent Configuration" on page 166
- "To Edit a Node Agent Configuration" on page 166
- "Creating a Node Agent" on page 169
- "Starting a Node Agent" on page 170
- "Stopping a Node Agent" on page 171
- "Deleting a Node Agent" on page 171

Working with Node Agents

▼ 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 170

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 169 and "Starting a Node Agent" on page 170

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.

See Also

- "Creating a Node Agent" on page 169
- "Starting a Node Agent" on page 170
- "Tasks Available through the Admin Console and asadmin Tool" on page 163
- "What Is a Node Agent?" on page 153
- "Node Agent Placeholders" on page 155
- "Node Agent and Domain Administration Server Synchronization" on page 158
- "To Edit a Node Agent Configuration" on page 166
- "To Delete a Node Agent Configuration" on page 166

▼ To Create a Node Agent Placeholder

Because the node agent must be created locally on the machine hosting the node agent, through the Admin Console you can only create 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 169.

For a list of the steps involved in creating and using node agents, see "Deploying Node Agents" on page 155

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`

See Also

- "Tasks Available through the Admin Console and asadmin Tool" on page 163
- "What Is a Node Agent?" on page 153

- "Node Agent Placeholders" on page 155
- "Creating a Node Agent" on page 169
- "Starting a Node Agent" on page 170
- "To Edit a Node Agent Configuration" on page 166
- "To Delete a Node Agent Configuration" on page 166

▼ 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 171.

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 171 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
```

See Also

- "Tasks Available through the Admin Console and asadmin Tool" on page 163
- "What Is a Node Agent?" on page 153
- "Node Agent Placeholders" on page 155
- "Stopping a Node Agent" on page 171
- "To Create a Node Agent Placeholder" on page 165
- "To Edit a Node Agent Configuration" on page 166
- "Deleting a Node Agent" on page 171

▼ 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 169.

   If this configuration is for an existing node agent, the node agent configuration information is synchronized automatically.

See Also

- "Tasks Available through the Admin Console and asadmin Tool" on page 163
- "What Is a Node Agent?" on page 153
- "Node Agent Placeholders" on page 155
- "Node Agent and Domain Administration Server Synchronization" on page 158
- "To Create a Node Agent Placeholder" on page 165
- "Creating a Node Agent" on page 169
- "Starting a Node Agent" on page 170
- "To Delete a Node Agent Configuration" on page 166

▼ 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.

See Also

- "What Is a Node Agent?" on page 153
- "Node Agent Placeholders" on page 155

- "To Edit a Node Agent Configuration" on page 166

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 more information, see "Working with Certificates and SSL" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*.

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.

See Also
- "What Is a Node Agent?" on page 153
- "Node Agent Placeholders" on page 155
- "To Edit a Node Agent Configuration" on page 166

Working with Node Agents using asadmin

You can perform the following node agent tasks with `asadmin`:

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.

EXAMPLE 7–1 Example of Creating a Node Agent

The following command creates a node agent:

```
asadmin create-node-agent --host myhost --port 4849 ---user admin nodeagent1
```

where *myhost* is the Domain Administration Server (DAS) hostname, `4849` is the DAS port number, `admin` is your DAS user, and `nodeagent1` is the name of the node agent being created.

Note – In the following situations, you must specify a DNS-reachable hostname:

- If domains cross subnet boundaries (that is, the node agent and the Domain Administration Server (DAS) are in different domains, for example, sun.com and java.com)
- If using a DHCP machine with a host name not registered in DNS.

Specify a DNS-reachable hostname by explicitly specifying the host name for the domain and the node agent when you create them:

```
create-domain --domainproperties domain.hostName=DAS-host-name
create-node-agent --hostDAS-host-name
--agentproperties remoteclientaddress=node-agent-host-name
```

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.

For more information, see:

- "What Is a Node Agent?" on page 153
- "Node Agent Placeholders" on page 155
- "Tasks Available through the Admin Console and asadmin Tool" on page 163
- "Deploying Node Agents" on page 155
- "To Create a Node Agent Placeholder" on page 165

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 nodeagent1
```

where *admin* is your administration user, and `nodeagent1` is the node agent being started.

For more information, see:

- "What Is a Node Agent?" on page 153
- "Node Agent Placeholders" on page 155
- "Tasks Available through the Admin Console and asadmin Tool" on page 163
- "Deploying Node Agents" on page 155
- "To Edit a Node Agent Configuration" on page 166

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.

For more information, see:

- "What Is a Node Agent?" on page 153
- "Deploying Node Agents" on page 155
- "Tasks Available through the Admin Console and asadmin Tool" on page 163
- "Starting a Node Agent" on page 170

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

For more information, see:

- "What Is a Node Agent?" on page 153
- "Deploying Node Agents" on page 155
- "Tasks Available through the Admin Console and asadmin Tool" on page 163
- "Stopping a Node Agent" on page 171

◆ ◆ ◆ CHAPTER 8

Configuring High Availability Session Persistence and Failover

This chapter explains how to enable and configure high availability session persistence:

Overview of Session Persistence and Failover

Application Server provides high availability session persistence through *failover* of HTTP session data and stateful session bean (SFSB) session data. Failover means that in the event of an server instance or hardware failure, another server instance takes over a distributed session.

Requirements

A distributed session can run in multiple Sun Java System Application Server instances, if:

- Each server instance has access to the same high-availability database (HADB). For information about how to enable this database, see `configure-ha-cluster`(1).
- Each server instance has the same distributable web application deployed to it. The `web-app` element of the `web.xml` deployment descriptor file must contain the `distributable` element.
- The web application uses high-availability session persistence. If a non-distributable web application is configured to use high-availability session persistence, the server writes an error to the log file.
- The web application must be deployed using the `deploy` or `deploydir` command with the `--availabilityenabled` option set to `true`. For more information on these commands, see `deploy`(1) and `deploydir`(1).

Restrictions

When a session fails over, any references to open files or network connections are lost. Applications must be coded with this restriction in mind.

You can only bind certain objects to distributed sessions that support failover. Contrary to the Servlet 2.4 specification, Sun Java System Application Server does not throw an `IllegalArgumentException` if an object type not supported for failover is bound into a distributed session.

You can bind the following objects into a distributed session that supports failover:

- Local home and object references for all EJB components.
- Co-located entity bean, stateful session bean, and distributed entity bean remote home reference, remote reference
- Distributed session bean remote home and remote references
- JNDI Context for `InitialContext` and `java:comp/env`.
- `UserTransaction` objects. However, if the instance that fails is never restarted, any prepared global transactions are lost and might not be correctly rolled back or committed.
- Serializable Java types

You cannot bind the following object types into sessions that support failover:

- JDBC DataSource
- Java Message Service (JMS) `ConnectionFactory` and `Destination` objects
- JavaMail™ Session
- Connection Factory
- Administered Objects.
- Web service reference

In general, for these objects, failover will not work. However, failover might work in some cases, if for example the object is serializable.

Sample Applications

The following directories contain sample applications that demonstrate session persistence:

install_dir`/samples/ee-samples/highavailability`
install_dir`/samples/ee-samples/failover`

The following sample application demonstrates SFSB session persistence:

install_dir`/samples/ee-samples/failover/apps/sfsbfailover`

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

Before You Begin High availability session persistence is incompatible with dynamic deployment, dynamic reloading, and auto-deployment. 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 Chapter 2, "Deploying Applications," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*.

1. **Create an Application Server cluster.**

   For more information, see "To Create a Cluster" on page 136 .

2. **Create an HADB database for the cluster.**

   For more information , see `configure-ha-cluster`(1).

3. **Set up HTTP load balancing for the cluster.**

   For more information , see "Setting Up HTTP Load Balancing" on page 105

4. **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 176.
   - Use the `asadmin` command-line utility. See `set`(1) and `configure-ha-persistence`(1).

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

   If the instance is currently serving requests, quiesce the instance before restarting it so that the instance gets enough time to serve the requests it is handling. For more information, see "Disabling (Quiescing) a Server Instance or Cluster" on page 121

6. **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 184

7. **Make each web module distributable if you want it to be highly available.**

8 **Enable availability for individual applications, web modules, or EJB modules during deployment.**

See "Configuring Availability for an Individual Application or EJB Module" on page 184

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. For instructions, see next section, "Enabling Availability for a Server Instance" on page 176 .
2. Container (web or EJB), enabled by default. For information on enabling availability at the container level, see:
   - "Configuring Availability for the Web Container" on page 177
   - "Configuring Availability for the EJB Container" on page 182
3. Application, disabled by default
4. Stand-alone web 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.

   Additionally, you can change the Store Pool Name if you changed the JDBC resource used for connections to the HADB for session persistence. For details, see `configure-ha-cluster`(1).
5. **Click on the Save button.**
6. **Stop and restart the server instance.**

HTTP Session Failover

J2EE 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

To enable and configure web container availability using `asadmin`, see `configure-ha-persistence`(1).

Alternatively, use the `asadmin set` command to set the configuration's `availability-service.web-container-availability.availability-enabled` property to true and then `configure-ha-persistence` to set properties as desired.

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.availability-enabled="true"
asadmin configure-ha-persistence --user admin --passwordfile secret.txt
--type ha
--frequency web-method
```

```
--scope modified-session
--store jdbc/hastore
--property maxSessions=1000:reapIntervalSeconds=60 cluster1
```

▼ 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,** **"Availability Settings" on page 178**
5. **Restart the server instance.**

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 `ha` (HADB).

HADB must be configured and enabled before you can use `ha` session persistence. For configuration details, see `configure-ha-cluster`(1).

If web container availability is enabled, the default is `ha`. Otherwise, the default is `memory`. For production environments that require session persistence, use `ha`. The first two types, memory and file persistence, do not provide high availability session persistence.

Persistence Frequency: Specifies how often the session state is stored. Applicable only if the Persistence Type is `ha`. 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 `ha`. 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 J2EE 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 180

HTTP Session Store: You can change the HTTP Session Store if you changed the JDBC resource used for connections to the HADB for session persistence. For details, see `configure-ha-cluster`(1).

Configuring Availability for Individual Web Applications

To enable and configure availability for an individual web 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 `ha` to enable high availability session persistence.

For more information about the `sun-web.xml` file, see "The sun-web.xml File" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Developer's Guide*.

Example

```
<sun-web-app> ...
  <session-config>
    <session-manager persistence-type=ha>
      <manager-properties>
        <property name=persistenceFrequency value=web-method />
      </manager-properties>
      <store-properties>
        <property name=persistenceScope value=session />
      </store-properties>
    </session-manager> ...
</session-config> ...
```

Using Single Sign-on with Session Failover

In a single application server instance, once a user is authenticated by an application, the user is not required to re-authenticate individually to other applications running on the same instance. This is called *single sign-on*. For more information, see "User Authentication for Single Sign-on" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Developer's Guide*.

For this feature to continue to work even when an HTTP session fails over to another instance in a cluster, single sign-on information must be persisted to the HADB. 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 tab of the Availability Service, as described in "Configuring Availability for the Web Container" on page 177 `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 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()`, 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 185

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 Application Server instance is stopped, the session data in the persistence store might not be cleared. To prevent this, undeploy the SFSB while the Application Server instance is running.

Configuring Availability for the EJB Container

▼ To Enable Availability for the EJB Container

1. **Select the EJB Container Availability tab.**
2. **Check the Availability Service box.**
   To disable availability, uncheck the box.
3. **Change other settings as described in "Availability Settings" on page 183**
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="ha"
```

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 `ha` (HADB). For production environments that require session persistence, use `ha`, the default.

SFSB Persistence Type: Specifies the passivation mechanism for SFSBs that *do not* have availability enabled. Allowed values are `file` (the default) and `ha`.

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 more information, see "To configure the store properties" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*.

HA persistence enables a cluster of server instances to recover the SFSB state if any server instance fails. HADB is also used as the passivation and activation store. Use this option in a production environment that requires SFSB state persistence. For more information, see `configure-ha-cluster`(1).

SFSB Store Pool Name: You can change the SFSB Store Pool Name if you changed the JDBC resource used for connections to the HADB for session persistence. For details, see `configure-ha-cluster`(1).

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 more information, see "To configure the store properties" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*..

Configuring Availability for an Individual Application or EJB Module

You can enable SFSB availability for an individual application or EJB module during deployment:

- If you are deploying with the Admin Console, check the Availability Enabled checkbox.
- If you are deploying using use the `asadmin deploy` or `asadmin deploydir` commands, set the `--availabilityenabled` option to `true`. For more information, see `deploy`(1) and `deploydir`(1).

Configuring Availability for an Individual Bean

To enable availability and select methods to be checkpointed for an individual SFSB, use the `sun-ejb-jar.xml` deployment descriptor file. .

To enable high availability session persistence, set `availability-enabled="true"` in the `ejb` element. To control the size and behavior of the SFSB cache, use the following elements:

- `max-cache-size` : specifies the maximum number of session beans that are held in cache. If the cache overflows (the number of beans exceeds `max-cache-size`), the container then passivates some beans or writes out the serialized state of the bean into a file. The directory in which the file is created is obtained from the EJB container using the configuration APIs.
- `resize-quantity`
- `cache-idle-timeout-in-seconds`
- `removal-timeout-in-seconds`
- `victim-selection-policy`

For more information about `sun-ejb-jar.xml`, see "The sun-ejb-jar.xml File" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Developer's Guide*

EXAMPLE 8–1 Example of an EJB Deployment Descriptor With Availability Enabled

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

EXAMPLE 8–1 Example of an EJB Deployment Descriptor With Availability Enabled *(Continued)*

```
        ...
        <ejb availability-enabled="true">
            <ejb-name>MySFSB</ejb-name>
        </ejb>
        ...
    </enterprise-beans>
</sun-ejb-jar>
```

Specifying Methods to Be Checkpointed

If enabled, checkpointing generally occurs after the bean completes any transaction, even if the transaction rolls back. To specify additional optional checkpointing of SFSBs at the end of non-transactional business methods that cause important modifications to the bean's state, use the `checkpoint-at-end-of-method` element in the `ejb` element of the `sun-ejb-jar.xml` deployment descriptor file.

The non-transactional methods in the `checkpoint-at-end-of-method` element can be:

- `create()` methods defined in the home interface of the SFSB, if you want to checkpoint the initial state of the SFSB immediately after creation
- For SFSBs using container managed transactions only, methods in the remote interface of the bean marked with the transaction attribute TX_NOT_SUPPORTED or TX_NEVER
- For SFSBs using bean managed transactions only, methods in which a bean managed transaction is neither started nor committed

  Any other methods mentioned in this list are ignored. At the end of invocation of each of these methods, the EJB container saves the state of the SFSB to persistent store.

Note – If an SFSB does not participate in any transaction, and if none of its methods are explicitly specified in the `checkpoint-at-end-of-method` element, the bean's state is not checkpointed at all even if `availability-enabled="true"` for this bean.

For better performance, specify a *small* subset of methods. The methods should accomplish a significant amount of work or result in important modification to the bean's state.

EXAMPLE 8–2 Example of EJB Deployment Descriptor Specifying Methods Checkpointing

```
<sun-ejb-jar>
    ...
    <enterprise-beans>
        ...
        <ejb availability-enabled="true">
```

EXAMPLE 8–2 Example of EJB Deployment Descriptor Specifying Methods Checkpointing *(Continued)*

```
            <ejb-name>ShoppingCartEJB</ejb-name>
            <checkpoint-at-end-of-method>
                <method>
                    <method-name>addToCart</method-name>
                </method>
            </checkpoint-at-end-of-method>
        </ejb>
        ...
    </enterprise-beans>
</sun-ejb-jar>
```

◆ ◆ ◆ CHAPTER 9

Java Message Service Load Balancing and Failover

This chapter describes how to configure load balancing and failover of the Java Message Service (JMS) for use with the Application Server. It contains the following topics:

Overview of Java Message Service

The Java Message Service (JMS) API is a messaging standard that allows J2EE 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 3 2005Q1 (MQ), which implements JMS, is tightly integrated with Application Server, enabling you to create components such as message-driven beans (MDBs).

MQ is integrated with Application Server using a *connector module,* also known as a resource adapter, defined by the J2EE Connector Architecture Specification 1.5. J2EE components deployed to the Application Server exchange JMS messages using the JMS provider integrated via the connector module. Creating a JMS resource in Application Server creates a connector resource in the background. So, each JMS operation invokes the connector runtime and uses the MQ resource adapter in the background.

You can manage the Java Message Service through the Admin Console or the `asadmin` command-line utility.

Sample Application

The `mqfailover` sample application demonstrates MQ failover with a Message Driven Bean receiving incoming messages from a JMS Topic. The sample contains an MDB and a

application client. The Application Server makes the MDB highly available. If one broker goes down, the conversational state (the messages received by MDB) is migrated transparently to another available broker instance in the cluster.

The sample is installed to

```
install_dir/samples/ee-samples/failover/apps/mqfailover
```

Further Information

For more information on JMS, see Chapter 14, "Using the Java Message Service," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Developer's Guide*. For more information on connectors (resource adapters), see Chapter 9, "Developing Connectors," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Developer's Guide*.

For more information about the Sun Java System Message Queue, see the Sun Java System Message Queue documentation. For general information about the JMS API, see the JMS web page (`http://java.sun.com/products/jms/index.html`)

Configuring the Java Message Service

The Java Message Service configuration is available to all inbound and outbound connections to the Sun Java System Application Server cluster or instance. You can configure the Java Message Service with:

- The Administration Console. Open the Java Message Service component under the relevant configuration. For details, see Chapter 4, "Configuring Java Message Service Resources," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*.
- The `asadmin set` command. You can set the following attributes:

```
server.jms-service.init-timeout-in-seconds = 60
server.jms-service.type = LOCAL
server.jms-service.start-args =
server.jms-service.default-jms-host = default_JMS_host
server.jms-service.reconnect-interval-in-seconds = 60
server.jms-service.reconnect-attempts = 3
server.jms-service.reconnect-enabled = true
server.jms-service.addresslist-behavior = random
server.jms-service.addresslist-iterations = 3
server.jms-service.mq-scheme = mq
server.jms-service.mq-service = jms
```

  You can also set these properties:

```
server.jms-service.property.instance-name = imqbroker
server.jms-service.property.instance-name-suffix =
server.jms-service.property.append-version = false
```

Use the `asadmin getcommand` to list all the Java Message Service attributes and properties. For more information on `asadmin get`, see `get`(1). For more information on `asadmin set`, see `set`(1).

You can override the Java Message Service configuration using JMS connection factory settings. For details, see "Admin Console Tasks for JMS Connection Factories" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*.

Note – You must restart the Application Server instance after changing the configuration of the Java Message Service.

For more information about JMS administration, see Chapter 4, "Configuring Java Message Service Resources," in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*

Java Message Service Integration

MQ can be integrated with Application Server in two ways: LOCAL and REMOTE, represented in Admin Console by the Java Message Service Type attribute.

LOCAL Java Message Service

When the Type attribute is LOCAL (the default for a stand-alone Application Server instances), the Application Server will start and stop the MQ broker specified as the Default JMS host. LOCAL type is most suitable for standalone Application Server instances.

To create a one-to-one relationship between Application Server instances and Message Queue brokers, set the type to LOCAL and give each Application Server instance a different default JMS host. You can do this regardless of whether clusters are defined in the Application Server or MQ.

With LOCAL type, use the Start Arguments attribute to specify MQ broker startup parameters.

REMOTE Java Message Service

When the Type attribute is REMOTE, the MQ broker must be started separately. This is the default if clusters are defined in the Application Server. For information about starting the broker, see the *Sun Java System Message Queue Administration Guide*.

In this case, Application Server will use an externally configured broker or broker cluster. Also, you must start and stop MQ brokers separately from Application Server, and use MQ tools to configure and tune the broker or broker cluster. REMOTE type is most suitable for Application Server clusters.

With REMOTE type, you must specify MQ broker startup parameters using MQ tools. The Start Arguments attribute is ignored.

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 Application Server uses.

The JMS Hosts list is populated with the hosts and ports of the specified MQ brokers and is updated whenever a JMS host configuration changes. When you create JMS resources or deploy MDBs, they inherit the JMS Hosts list.

Note – In the Sun Java System Message Queue software, the `AddressList` property is called `imqAddressList`.

Default JMS Host

One of the hosts in the JMS Hosts list is designated the default JMS host, named `Default_JMS_host`. The Application Server instance starts the default JMS host when the Java Message Service type is configured as LOCAL.

If you have created a multi-broker cluster in the Sun Java System Message Queue software, delete the default JMS host, then add the Message Queue cluster's brokers as JMS hosts. In this case, the default JMS host becomes the first one in the JMS Hosts list.

When the Application Server uses a Message Queue cluster, it executes Message Queue specific commands on the default JMS host. For example, when a physical destination is created for a Message Queue cluster of three brokers, the command to create the physical destination is executed on the default JMS host, but the physical destination is used by all three brokers in the cluster.

Creating JMS Hosts

You can create additional JMS hosts in the following ways:

- Use the Administration Console. Open the Java Message Service component under the relevant configuration, select the JMS Hosts component, and then click New. For more information, see "To create a JMS host" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*.

- 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

Application Server supports JMS connection pooling and failover. The Sun Java System Application Server pools JMS connections automatically. When the Address List Behavior attribute is `random` (the default), Application Server selects its primary broker randomly from the JMS host list. When failover occurs, MQ transparently transfers the load to another broker and maintains JMS semantics.

To specify whether the Application Server tries to reconnect to the primary broker when the connection is lost, select the Reconnect checkbox. If enabled and the primary broker goes down, Application Server tries to reconnect to another broker in the JMS Hosts list.

When Reconnect is enabled, also specify the following attributes:

- **Address List Behavior**: whether connection attempts are in the order of addresses in the JMS Hosts List (priority) or random order (random). If set to Priority, Java Message Service tries to connect to the first MQ broker specified in the JMS Hosts list and uses another one only if the first broker is not available. If set to Random, Java Message Service selects the MQ broker randomly from the JMS Hosts list. If there are many clients attempting a connection using the same connection factory, use this setting to prevent them from all attempting to connect to the same address.
- **Address List Iterations**: number of times the Java Message Service iterates through the JMS Hosts List in an effort to establish (or re-establish) a connection). A value of -1 indicates that the number of attempts is unlimited.
- **Reconnect Attempts**: the number of attempts to connect (or reconnect) for each address in the JMS hosts list before the client runtime tries the next address in the list. A value of -1 indicates that the number of reconnect attempts is unlimited (the client runtime attempts to connect to the first address until it succeeds).
- **Reconnect Interval**: number of seconds between reconnect attempts. This applies for attempts on each address in the JMS hosts list and for successive addresses in the list. If it is too short, this time interval does not give a broker time to recover. If it is too long, the reconnect might represent an unacceptable delay.

You can override these settings using JMS connection factory settings. For details, see "Admin Console Tasks for JMS Connection Factories" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Administration Guide*.

Load-Balanced Message Inflow

Application Server delivers messages randomly to MDBs having 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 MDB that subscribe to same topic are considered equal. When an MDB is deployed to multiple instances of the Application Server, only one of the MDBs receives the message. If multiple distinct MDBs subscribe to same topic, one instance of each MDB 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, MQ allows up to that number of MDBs to consume messages from same queue. The message is delivered randomly to the MDBs. If `maxNumActiveConsumers` is set to `-1`, there is no limit to the number of consumers.

Using MQ Clusters with Application Server

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

This section describes how to configure Application Server to use highly available Sun Java System Message Queue clusters. It explains how to start and configure Message Queue clusters.

For more information about the topology of Application Server and MQ deployment, see "Planning Message Queue Broker Deployment" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Deployment Planning Guide*.

▼ To Enable MQ clusters with Application Server Clusters

1 Create an Application Server cluster, if one does not already exist.

For information on creating clusters, see "To Create a Cluster" on page 136.

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 hosts with either the Admin Console or the `asadmin` command-line utility.

To use `asadmin`, the commands are for example:

```
asadmin delete-jms-host --target cluster1 default_JMS_host
asadmin create-jms-host --target cluster1
```

```
     --mqhost myhost1 --mqport 6769
     --mquser admin --mqpassword admin broker1
asadmin create-jms-host --target cluster1
     --mqhost myhost2 --mqport 6770
     --mquser admin --mqpassword admin broker2
asadmin create-jms-host --target cluster1
     --mqhost myhost3 --mqport 6771
     --mquser admin --mqpassword admin broker3
```

To create the hosts with Admin Console:

a. **Navigate to the JMS Hosts node (Configurations > *config-name* > Java Message Service > JMS Hosts)**

b. **Delete the default broker (default_JMS_host).**

   Select the checkbox next to it, and then click Delete.

c. **Click New to create each JMS host and enter its property values.**

   Fill in the values for host name, DNS name or IP address, port number, administrative user name and password.

3 **Start the master MQ broker and the other MQ brokers.**

In addition to the three external brokers started on JMS host machines, start one master broker on any machine. This master broker need not be part of a broker cluster. For example:

```
/usr/bin/imqbrokerd -tty -name brokerm -port 6772
 -cluster myhost1:6769,myhost2:6770,myhost2:6772,myhost3:6771
 -D"imq.cluster.masterbroker=myhost2:6772"
```

4 **Start the Application Server instances in the cluster.**

5 **Create JMS resources on the cluster:**

a. **Create JMS physical destinations.**

   For example, using `asadmin`:

   ```
   asadmin create-jmsdest --desttype queue --target cluster1 MyQueue
   asadmin create-jmsdest --desttype queue --target cluster1 MyQueue1
   ```

   To use Admin Console:

   i. **Navigate to the JMS Hosts page (Configurations > *config-name* > Java Message Service > Physical Destinations).**

   ii. **Click New to create each JMS physical destination.**

   iii. **For each destination, enter its name and type (queue).**

b. Create JMS connection factories.

For example, using `asadmin`:

```
asadmin create-jms-resource --target cluster1
--restype javax.jms.QueueConnectionFactory jms/MyQcf
asadmin create-jms-resource --target cluster1
--restype javax.jms.QueueConnectionFactory jms/MyQcf1
```

To use Admin Console:

i. Navigate to the JMS Connection Factories page (Resources > JMS Resources > Connection Factories).

ii. To create each connection factory, click New.

The Create JMS Connection Factory page opens.

iii. For each connection factory, enter JNDI Name (for example `jms/MyQcf`) and Type, `javax.jms.QueueConnectionFactory`

iv. Select the cluster from the list of available targets at the bottom of the page and click Add.

v. Click OK to create the connection factory.

c. Create JMS destination resources.

For example, using `asadmin`:

```
asadmin create-jms-resource --target cluster1
     --restype javax.jms.Queue
     --property imqDestinationName=MyQueue jms/MyQueue
asadmin create-jms-resource --target cluster1
     --restype javax.jms.Queue
     --property imqDestinationName=MyQueue1 jms/MyQueue1
```

To use Admin Console:

i. Navigate to the JMS Destination Resources page (Resources > JMS Resources > Connection Factories).

ii. To create each destination resource, click New.

The Create JMS Destination Resource page opens.

iii. For each destination resource, enter JNDI Name (for example `jms/MyQueue`) and Type `javax.jms.Queue`.

iv. Select the cluster from the list of available targets at the bottom of the page and click Add.

v. **Click OK to create the destination resource.**

6 **Deploy the applications with the** – `retrieve` **option for application clients. For example:**

```
asadmin deploy --target cluster1
--retrieve /opt/work/MQapp/mdb-simple3.ear
```

7 **Access the application and test it to ensure it is behaving as expected.**

8 **If you want to return the Application Server to its default JMS configuration, delete the JMS hosts you created and recreate the default. For example:**

```
asadmin delete-jms-host --target cluster1 broker1
asadmin delete-jms-host --target cluster1 broker2
asadmin delete-jms-host --target cluster1 broker3
asadmin create-jms-host --target cluster1
 --mqhost myhost1 --mqport 7676
 --mquser admin --mqpassword admin
 default_JMS_host
```

You can also perform the equivalent operation with Admin Console.

Troubleshooting If you encounter problems, consider the following:

- View the Application Server log file. If you see in the log file that an MQ broker does not respond to a message, stop the broker and then restart it.
- Always be sure to start MQ brokers first, then Application Server instances.
- When all MQ brokers are down, it takes 30 minutes for Application Server to go down or up, with the default values in Java Message Service. Tune Java Message Service values to get acceptable values for this timeout. For example:

```
asadmin set --user admin --password administrator
cluster1.jms-service.reconnect-interval-in-seconds=5
```

◆ ◆ ◆ CHAPTER 10

RMI-IIOP Load Balancing and Failover

This chapter describes using Sun Java System Application Server's high-availability features for remote EJB references and JNDI objects over RMI-IIOP.

Overview

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. However, adding or deleting new instances to the cluster will not update an existing client\qs view of the cluster. To do so, you must manually update the endpoints list on the client side.

Requirements

Sun Java System Application Server Enterprise Editionprovides high availability of remote EJB references and `NameService` objects over RMI-IIOP, provided all the following apply:

- Your deployment has a cluster of at least two application server instances.
- J2EE applications are deployed to all application server instances and clusters that participate in load balancing.
- RMI-IIOP client applications are enabled for load balancing.

Application Server supports load balancing for the following RMI-IIOP clients accessing EJB components deployed on an Application Server.

- Java applications executing in the Application Client Container (ACC). See "To set up RMI-IIOP load balancing for the Application Client Container" on page 199.
- Java applications not running in the ACC. See "To set up RMI-IIOP load balancing and failover for Stand-Alone Client" on page 201

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

Algorithm

Application Server uses a randomization and round-robin algorithm for RMI-IIOP load balancing and failover.

When an RMI-IIOP client first creates a new `InitialContext` object, the list of available Application Server IIOP endpoints is randomized for that client. For that `InitialContext` object, the load balancer directs lookup requests and other `InitialContext` operations to the first endpoint on the randomized list. If the first endpoint is not available then the second endpoint in the list is used, and so on.

Each time the client subsequently creates a new `InitialContext` object, the endpoint list is rotated so that a different IIOP endpoint is used for `InitialContext` operations.

When you obtain or create beans from references obtained by an `InitialContext` object, those beans are created on the Application Server instance serving the IIOP endpoint assigned to the `InitialContext` object. The references to those beans contain the IIOP endpoint addresses of all Application Server instances in the cluster.

The *primary endpoint* is the bean endpoint corresponding to the `InitialContext` endpoint used to look up or create the bean. The other IIOP endpoints in the cluster are designated as *alternate endpoints*. If the bean\qs 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 and with standalone Java clients.

Sample Application

The following directory contains a sample application that demonstrates using RMI-IIOP failover with and without ACC:

install_dir`/samples/ee-samples/sfsbfailover`

See the `index.html` file accompanying this sample for instructions on running the application with and without ACC. The `ee-samples` directory also contains information for setting up your environment to run the samples.

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) and for standalone client applications.

▼ To set up RMI-IIOP load balancing for the Application Client Container

This procedure gives an overview of the steps necessary to use RMI-IIOP load balancing and failover with the application client container (ACC). For additional information on the ACC, see "Developing Clients Using the ACC" in *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 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 numbers for the instances in the cluster.**

Specify the IIOP listeners as endpoints to determine which IIOP listener receives the requests. To display the IIOP listeners in the Admin Console:

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.**

Note the IIOP listener port for the specific instance.

e. **Repeat the process for every instance.**

7 **Edit `sun-acc.xml` for the endpoint values.**

Using the IIOP Listeners from the previous step, create endpoint values in the form:

machine1:*instance1–iiop-port*, *machine2*:*instance2–iiop-port*

For example:

```
<property name="com.sun.appserv.iiop.endpoints"
value="host1.sun.com:3335,host2.sun.com:3333,host3.sun.com:3334"\>
```

8 **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
```

9 **Run the application client as follows:**

`appclient -client` *clientjar* `-name` *appname*

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.

▼ To set up RMI-IIOP load balancing and failover for Stand-Alone Client

1. **Deploy the 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
   ```

2. **Run the client jar and the required jar files, specifying the endpoints and `InitialContext` as `-D` values.**

   For example:

   ```
   java -Dcom.sun.appserv.iiop.endpoints=
   host1.sun.com:33700,host2.sun.com:33700,host3.sun.com:33700
   samples.rmiiiopclient.client.Standalone_Client
   ```

Next Steps To test failover, stop one instance in the cluster and confirm 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

E

F

G

H

I

J

L

T

U

V

W