Sun Java™ System

Application Server Enterprise Edition 8.1 High Availability Administration Guide

2005Q1

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

Part Number 819-0216

Contents

Preface

This guide describes how to configure and administer the Application Server High Availability database. This preface contains information about the following topics:

Who Should Use This Book

This *High Availability Administration Guide* is intended for information technology administrators in production envrionments. This guide assumes you are familiar with the following topics:

- Basic system administration tasks
- Installing software
- Using Web browsers
- Starting database servers

- Issuing commands in a terminal window

Before You Read This Book

Application Server is a component of Sun Java™ Enterprise System, a software infrastructure that supports enterprise applications distributed across a network or Internet environment. You should be familiar with the documentation provided with Sun Java Enterprise System, which can be accessed online at `http://docs.sun.com/coll/entsys.05q1#hic`.

How This Book Is Organized

The organization of this guide corresponds to the layout of the Admin Console, the browser-based tool for administering the Application Server. Each chapter begins with conceptual information, followed by procedural sections that explain how to perform specific tasks with the Admin Console.

Conventions Used in This Book

The tables in this section describe the conventions used in this book.

Typographic Conventions

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

Table 1 Typographic Conventions

Typeface	Meaning	Examples
AaBbCc123 (Monospace)	API and language elements, HTML tags, web site URLs, command names, file names, directory path names, onscreen computer output, sample code.	Edit `your.login` file. Use `ls -a` to list all files. `% You have mail.`
AaBbCc123 (Monospace bold)	What you type, when contrasted with onscreen computer output.	`% su` `Password:`

Table 1 Typographic Conventions *(Continued)*

Typeface	Meaning	Examples
AaBbCc123 (Italic)	Book titles, new terms, words to be emphasized. A placeholder in a command or path name to be replaced with a real name or value.	Read Chapter 6 in the *User's Guide*. These are called *class* options. Do *not* save the file. The file is located in the *install-dir*`/bin` directory.

Symbols

The following table describes the symbol conventions used in this book.

Table 2 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional 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.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
>	Indicates menu item selection in a graphical user interface.	File > New > Templates	From the File menu, choose New. From the New submenu, choose Templates.

Default Paths and File Names

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

Table 3 Default Paths and File Names

Term	Description

install_dir	By default, the Application Server installation directory is located here:
	• 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	By default, the directory containing all domains is located here:
	• 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	By default, each domain directory is located here: *domain_root_dir*`/`*domain_dir* In configuration files, you might see *domain_dir* represented as follows: `${com.sun.aas.instanceRoot}`

Shell Prompts

The following table describes the shell prompts used in this book.

Table 4 Shell Prompts

Shell	Prompt
C shell on UNIX or Linux	*machine-name*%
C shell superuser on UNIX or Linux	*machine-name*#
Bourne shell and Korn shell on UNIX or Linux	$
Bourne shell and Korn shell superuser on UNIX or Linux	#
Windows command line	C:\

Related Documentation

The http://docs.sun.com[SM] web site enables you to access Sun technical documentation online. You can browse the archive or search for a specific book title or subject.

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

General J2EE Information:

The J2EE 1.4 Tutorial:

http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

The J2EE Blueprints:

http://java.sun.com/reference/blueprints/index.html

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

Java Security, by Scott Oaks, O'Reilly Publishing

Programming with Servlets and JSP files:

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

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

Programming with EJB components:

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

Programming with JDBC:

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

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

Books in This Documentation Set

The Sun Java System Application Server manuals are available as online files in Portable Document Format (PDF) and Hypertext Markup Language (HTML).

The following table summarizes the books included in the Application Server core documentation set.

Table 5 Books in This 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.
Quick Start Guide	How to get started with the Sun Java System Application Server product.
Installation Guide	Installing the Sun Java System Application Server software and its components.
Developer's Guide	Creating and implementing Java™ 2 Platform, Enterprise Edition (J2EE™ platform) applications intended to run on the Sun Java System 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 and deploying the applications on the Sun Java System Application Server.
Administration Guide	Configuring, managing, and deploying the Sun Java System Application Server subsystems and components from the Administration Console.
Administration Reference	Editing the Sun Java System Application Server configuration file, `domain.xml`.
Upgrade and Migration Guide	Migrating your applications to the new Sun Java System 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.
Troubleshooting Guide	Solving Sun Java System Application Server problems.
Error Message Reference	Solving Sun Java System Application Server error messages.

Table 5 Books in This Documentation Set *(Continued)*

Book Title	Description
Reference Manual	Utility commands available with the Sun Java System Application Server; written in manpage style. Includes the `asadmin` command line interface.

Other Server Documentation

For other server documentation, go to the following:

- Message Queue documentation
  http://docs.sun.com/db?p=prod/s1.s1msgqu
- Directory Server documentation
  http://docs.sun.com/coll/DirectoryServer_04q2
- Web Server documentation
  http://docs.sun.com/coll/S1_websvr61_en

Accessing Sun Resources Online

For product downloads, professional services, patches and support, and additional developer information, go to the following:

- Download Center
  http://wwws.sun.com/software/download/
- Professional Services
  http://www.sun.com/service/sunps/sunone/index.html
- Sun Enterprise Services, Solaris Patches, and Support
  http://sunsolve.sun.com/
- Developer Information
  http://developers.sun.com/prodtech/index.html

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in the product documentation, go to http://www.sun.com/service/contacting.

Related Third-Party Web Site References

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 document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the title of this book is *Sun Java System Application Server 2005Q1 High Availability Administration Guide*, and the part number is 817-6088.

Chapter 1

Application Server High Availability Features

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

- Overview of High Availability
- HTTP Load Balancing and Failover
- High Availability Session Persistence
  - HTTP Session Failover
  - Stateful Session Bean Failover
- RMI-IIOP Load Balancing and Failover
- Java Message Service Load Balancing and Failover

Overview of High Availability

High availability applications and services provide their functionality continuously, regardless of hardware and software failures. Application Server provides high availability through the following sub-components and features:

- High Availability Database (HADB)
- Load Balancer Plug-in
- Highly Available Clusters

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. Generally, you must install, configure, and manage HADB independently of Application Server. HADB is designed to support up to 99.999% service and data availability with load balancing, failover, and state recovery.

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" chapter and the "Selecting a Topology" chapter in the *Sun Java System Application Server Enterprise Edition Deployment Guide.*

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.

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 "HTTP Load Balancing and Failover" on page 21.

Highly Available Clusters

A highly available cluster in the Sun Java System Application Server Enterprise Edition integrates a state replication service with clusters and load balancer. A cluster is a collection of server instances that work together as one logical entity. A cluster provides a runtime environment for one or more J2EE applications.

Using Application Server 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 of the Application Server 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, 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" in the *Sun Java System Application Server Administration Guide*.
- 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 more information about

- Administering high availability features, see the *Sun Java System Application Server Administration Guide*.
- Developing and deploying applications that take advantage of high availability features, see the *Sun Java System Application Server Developer's Guide*.

Planning a High Availability Deployment

For information about planning a high-availability deployment, including assessing hardware requirements, planning network configuration, and selecting a topology, see *Sun Java System Application Sever 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

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 the chapter "Tuning for High-Availability" in the *Sun Java System Application Server Performance Tuning Guide*. This manual 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

HTTP Load Balancing and Failover

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

- Upgrading Applications Without Loss of Availability

For more information about the HTTP load balancer plug-in, including monitoring the load balancer, See the chapter “Configuring Load Balancing and Failover” in the *Sun Java System Application Server Administration Guide.*

How the Load Balancer Works

- Assigned Requests and Unassigned Requests
- HTTP Load Balancing Algorithm
- Sample Applications

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 pure round robin.

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-Based 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 Method: 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.

Prerequisites

To setup a system with load balancing, in addition to the Application Server, you must install a web server and the load-balancer plug-in. Then you must:

- Create Application Server clusters that you want to participate in load balancing.
- Deploy applications to these load-balanced clusters.
- Configure the web server. For more information, see “Configuring Web Servers for HTTP Load Balancing” on page 24.

Server instances and clusters participating in load balancing must have a homogenous environment. Usually that means that the server instances reference the same server configuration and have the same applications deployed to them.

Procedure to Set Up Load Balancing

Use the `asadmin` tool to configure load balancing in your environment. Follow these steps:

1. Create a load balancer configuration using the `asadmin` command `create-http-lb-config`.
2. Add references to clusters and stand-alone server instances 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 clusters or stand-alone server instances reference 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 HTTP 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 *Sun Java System Application Server Installation Guide.*

Modifications to Sun Java System Web Server

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

1. Adds the following load balancer plug-in specific entries to the web server instance's `magnus.conf` file:

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

Init fn="init-passthrough"

##end addition for EE lb-plugin
```

2. Adds the following entries specific to the load balancer plug-in to the web server instance's `obj.conf` file:

```
<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>
```

`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 23.

Modifications to Apache Web Server

For the Apache Web Server, your installation must meet the minimum requirements, and you must perform certain configuration steps before installing the Sun Java System Application Server load balancer plug-in.

Additional modifications to the Apache Web Server are made during the load balancer plug-in installation. After the plug-in is installed, additional configuration is required.

- Configuration before Installing the Load Balancer Plug-in
- Modifications Made by the Application Server Installer
- Modifications After Installation

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.

Minimum Requirements for Apache 1.3:

- `openssl-0.9.7e` (source)
- `mod_ssl-2.8.16-1.3.x` (source) 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)

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 `/usr/local/lib/gcc-lib/sparc-sun-solaris2.9/3.3/install-tools/mkheaders` (Solaris SPARC) or `/usr/local/lib/gcc-lib/i386-pc-solaris2.9/3.3/install-tools/mkheaders` (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 another C compiler, set the path of the C compiler and make utility in the PATH environment variable. For example:

  ```
  exportLD_LIBRARY_PATH=$LD_LIBRARY_PATH:appserver_installdir/lib
  ```

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

Minimum Requirements for Apache 2:

- `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)

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 `/usr/local/lib/gcc-lib/sparc-sun-solaris2.9/3.3/install-tools/mkheaders` (Solaris SPARC) or `/usr/local/lib/gcc-lib/i386-pc-solaris2.9/3.3/install-tools/mkheaders` (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 another C compiler, set the path of the C compiler and make utility in the PATH environment variable. For example:

  ```
  export LD_LIBRARY_PATH=app_server_install_dir/lib:$LD_LIBRARY_PATH
  ```

  This example is for sh.
- These software sources are available at http://www.sunfreeware.com

Installing SSL-aware Apache:

You must have already downloaded and uncompressed the Apache software before starting.

1. Compile and build OpenSSL. For more information about OpenSSL, see:

   http://www.openssl.org/

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

   Download and unpack the OpenSSL source.

   a. `cd openssl-0.9.7e`

   b. `make`

   c. `make install`

   For Apache 1.3, configure Apache with mod_ssl. You do not need to complete this step for Apache 2. For more information about mod_ssl, see:

   http://www.modssl.org/

   Unpack the mod_ssl source and follow these steps.

   a. `cd mod_ssl-2.8.14-1.3.x`

   b. Run `./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`

   The directory specified in the above command example is a variable. The *prefix* argument indicates where to install Apache. The *x* in the version number represents your actual version.

2. For Apache 2.0, configure the source tree:

   a. cd to the `http-2.0_x` directory.

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

   The directory specified in the above command example is a variable. The *prefix* argument indicates where to install Apache. The *x* in the version number represents your actual version.

3. 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. Create an environment variable LD_LIBRARY_PATH equal to: *appserver_install_dir*`/lib` (for all installations) and *appserver_install_dir*`/lib:opt/sun/private/lib` (for Application Server installed as part of a Java Enterprise System installation).

4. Compile Apache as described in the installation instructions for the version you are using. Full documentation is at:

   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.

5. Configure Apache for your environment.

Modifications Made by the Application Server Installer

The installation program installing the load balancer plug-in extracts the necessary files to the `libexec` (Apache 1.3) or `modules` (Apache 2.0) folder under the web server's root directory. It adds the following entries specific to the load balancer plug-in 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
```

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.

Modifications After Installation

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`.

Additional Modifications on the Solaris Platform

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`.

Additional Modifications on the Linux Platform

On the Linux platform, 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`.

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.

1. Configure the new web server instance to use the load balancer plug-in, as described in "Modifications to Sun Java System Web Server" on page 25 or "Modifications to Apache Web Server" on page 26.
2. Copy the `sun-loadbalancer_1_1.dtd` file from the existing web server instance's `config` directory to the new instance's `config` directory.
3. To use the same load balancer configuration, copy the `loadbalancer.xml` file from the existing web server instance's `config` directory to the new instance's `config` directory.
4. To use a different load balancer configuration:
   a. Create a new load balancer configuration using `asadmin create-http-lb-config`.
   b. Export the new configuration to a `loadbalancer.xml` file using `asadmin export http-lb-config`.
   c. Copy that `loadbalancer.xml` file to the new web server's `config` directory.

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

Configuring the Load Balancer

A load balancer configuration is a named configuration in the `domain.xml` file that defines a load balancer. 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:

- Changing the HTTP Load Balancer Configuration
- Enabling Dynamic Reconfiguration
- Disabling (Quiescing) a Server Instance or Cluster
- Disabling (Quiescing) an Application

Creating an HTTP Load Balancer Configuration

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

Table 1-1 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 HTTP and HTTPS Session Failover" on page 39.
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 37.
monitor	Whether monitoring is enabled for the load balancer. For more information, see the *Sun Java System Application Server Administration Guide*.
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-checker` command. Specify the following parameters:

- url

  Specifies the listener's URL that the load balancer checks to determine its state of health. The default is `"/"`.

- interval

  Specifies the interval in seconds at which health checks of instances occur. The default is 30 seconds. Specifying 0 disables the health checker.

- timeout

  Specifies the timeout interval in seconds within which a response must be obtained for a listener to be considered healthy. The default is `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 1-2 Health-checker 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:

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.

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 HTTP Load Balancer Configuration

If you change an HTTP 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 36.

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

When dynamic reconfiguration is enabled, the load balancer plug-in periodically checks for an updated configuration. To enable dynamic reconfiguration:

- To enable dynamic reconfiguration when creating a load balancer configuration, use the `--reloadinterval` option when running `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 reloading. By default, dynamic reloading is enabled, and the interval is set to 60 seconds.

- To enable dynamic reloading if you have disabled it, or to change the reload interval, use the `asadmin set` command.

After changing these settings, export the load balancer configuration file again and copy it to the web server's `config` directory.

If you enable dynamic reconfiguration after it has previously been disabled, you also must 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.

The load balancer uses the following policy for quiescing applications:

- If an application is disabled, and the timeout has not expired, new requests to the disabled applications are not forwarded by the load balancer. These requests are returned to the web server. Sticky requests continue to be forwarded until the timeout expires.
- When the timeout expires, the application is disabled. The load balancer does not accept any requests for this 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. Run `asadmin disable-http-lb-application`, specifying the timeout (in minutes) the name of the application to disable, and the 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 Session 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
- Configuring Idempotent URLs

HTTPS Routing

All incoming requests, whether HTTP or HTTPS, are routed by the load balancer plug-in to application server instances. However, if HTTPS routing is enabled, a HTTPS request will be forwarded by the load balancer plug-in to the application server using an HTTPS port only. Note that 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

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

Known Issues

The following list discusses the limitations in Load Balancer with respect to 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 is valid regardless of the value set for `httpsrouting`.

Configuring Idempotent URLs

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.

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.

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 the *Sun Java System Application Server 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 42 and "Upgrading in Multiple Clusters" on page 44

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 45.

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 `asadmin` 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:
   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:
   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 Step c of Step 5.
9. Repeat Step 5 through Step c for each instance in the cluster.
10. When all server instances have the new application and are running, enable dynamic reconfiguration for the cluster again.

Upgrading in Multiple Clusters

Follow these steps to upgrade a compatible application deployed 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.

   Through the 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 `asadmin` 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 Step 5 through Step 8 for the other clusters.
10. When all server instances have the new application and are running, enable dynamic reconfiguration for all clusters again.

Upgrading Incompatible Applications

For information on what makes applications compatible, see "Application Compatibility" on page 41. You must use a different rolling upgrade procedure if 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.

High Availability Session Persistence

This chapter explains how to enable and configure high availability session pleasantness:

- Overview of Session Failover
- Setting Up High Availability Session Persistence
- Enabling Session Availability
- HTTP Session Failover
- Stateful Session Bean Failover

Overview of Session 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 the description of the `configure-ha-cluster` command in the *Reference Manual*.
- 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, an error is written to the server log.
- The web application must be deployed using the `deploy` or `deploydir` command with the `--availabilityenabled` option set to `true`. See the *Sun Java System Application Server Reference Manual*.

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 Object.
- 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

To enable high availability session persistence, follow these steps:

1. Create an Application Server cluster. For more information, see the chapter "Configuring Clusters" in the *Sun Java System Application Server Administration Guide*.
2. Create an HADB database for the cluster. See the description of the `configure-ha-cluster` command in the *Reference Manual*.
3. Set up HTTP load balancing for the cluster. See "HTTP Load Balancing and Failover" on page 21.
4. Enable availability for the application server instances and web or EJB containers that you want to support high availability session persistence, and configure the session persistence settings. Choose one of these approaches:
   - See "Enabling Availability for a Server Instance" on page 51.
   - See the description of the `configure-ha-persistence` command in the *Reference Manual*.
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 38.

6. Enable availability for any specific SFSB that requires it, and select methods for which checkpointing the session state is necessary. See "Configuring Availability for an Individual Bean" on page 57 and "Specifying Methods to Be Checkpointed" on page 58.
7. Make each web module distributable if you want it to be highly available. For more information, see the *Sun Java System Application Server Developer's Guide*.
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 57, "Configuring Availability for Individual Web Applications" on page 53.

   In the Administration Console, check the Availability Enabled box, or use the `deploy` command with the `--availabilityenabled` option set to `true`.

NOTE High availability session persistence is incompatible with dynamic deployment, dynamic reloading, and auto-deployment. These features are for development, not production environments. For information about how to disable these features, see the *Sun Java System Application Server Administration Guide*.

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".
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 51
   - "Configuring Availability for the EJB Container" on page 56
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 using the Administration Console:

1. In the tree component, expand the Configurations node.
2. Expand the node for the configuration you want to edit.
3. 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 the description of the `configure-ha-cluster` command in the *Reference Manual*.
4. Click on the Save button.
5. 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 the Administration 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". Click on the Save button.

5. Restart the server instance.

To enable and configure web container availability using `asadmin`, see the `configure-ha-persistence` command in the *Reference Manual*.

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` (the HADB).

HADB must be configured and enabled before you can use `ha` session persistence. For configuration details, see the description of the `configure-ha-cluster` command in the *Reference Manual*.

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; for more information on them, see the *Sun Java System Application Server Developer's Guide*.

Persistence Frequency: Specifies how often the session state is stored. Applicable only if the Persistence Type is `ha`. Allowed values are as follows:

- `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 54.

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 the description of the `configure-ha-cluster` command in the *Reference Manual*.

Configuring Availability for Individual Web Applications

You can enable availability and configure availability settings for an individual web application, in its `sun-web.xml` deployment descriptor file. 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 Java System Application Server 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 reauthenticate individually to other applications running on the same instance. This is called *single sign-on*. For more information on single sign-on, see the chapter "Configuring Security" in the *Sun Java System Application Server 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. First enable availability for the server instance and the web container, then enable single-sign-on state persistence. See "Enabling Availability for a Server Instance" on page 51.

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 58.

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 Sun Java System Application Server instance is stopped, the session data in the persistence store might not be cleared. To prevent this, undeploy the SFSB while the Sun Java System Application Server instance is running.

Configuring Availability for the EJB Container

To enable availability for the EJB container using the Administration Console:

1. Select the EJB Container Availability tab, then check the Availability Service box. To disable it, uncheck the box.
2. Change other settings, as described in the following section, "Availability Settings".
3. Click on the Save button.
4. Restart the server instance.

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` (the 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.

HA persistence enables a cluster of server instances to recover the SFSB state if any server instance fails. The HADB is also used as the passivation and activation store. Use this option in a production environment that requires SFSB state persistence. For information about how to set up and configure this database, see the description of the `configure-ha-cluster` command in the *Reference Manual*.

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 the description of the `configure-ha-cluster` command in the *Reference Manual*.

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 the chapter "J2EE Containers" in the *Sun Java System Application Server 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 details, type `man deploy` or `man deploydir`, or see the *Sun Java System Application Server Reference Manual*.

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. For details, see the *Sun Java System Application Server Developer's Guide.*

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`
- `resize-quantity`
- `cache-idle-timeout-in-seconds`
- `removal-timeout-in-seconds`
- `victim-selection-policy`

The max-cache-size element specifies the maximum number of session beans that are held in cache. If the cache overflows (when 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.

For more information about sun-ejb-jar.xml, see the appendix "Deployment Descriptor Files" in the *Sun Java System Application Server Developer's Guide.*

Example

```
<sun-ejb-jar>
    ...
    <enterprise-beans>
        ...
        <ejb availability-enabled="true">
            <ejb-name>MySFSB</ejb-name>
        </ejb>
        ...
    </enterprise-beans>
</sun-ejb-jar>
```

Specifying Methods to Be Checkpointed

If enabled, checkpointing generally occurs after the bean completes any transaction, even if the transaction rolls back. To specify additional optional checkpointing of SFSBs at the end of non-transactional business methods that cause important modifications to the bean's state, use the checkpoint-at-end-of-method element in the ejb element of the sun-ejb-jar.xml deployment descriptor file.

The non-transactional methods in the checkpoint-at-end-of-method element can be:

- create() methods defined in the home interface of the SFSB, if you want to checkpoint the initial state of the SFSB immediately after creation
- For SFSBs using container managed transactions only, methods in the remote interface of the bean marked with the transaction attribute TX_NOT_SUPPORTED or TX_NEVER
- For SFSBs using bean managed transactions only, methods in which a bean managed transaction is neither started nor committed

Any other methods mentioned in this list are ignored. At the end of invocation of each of these methods, the EJB container saves the state of the SFSB to persistent store.

NOTE If an SFSB does not participate in any transaction, and if none of its methods are explicitly specified in the `checkpoint-at-end-of-method` element, the bean's state is not checkpointed at all even if `availability-enabled="true"` for this bean.

For better performance, specify a *small* subset of methods. The methods should accomplish a significant amount of work or result in important modification to the bean's state.

For example:

```
<sun-ejb-jar>
    ...
    <enterprise-beans>
        ...
        <ejb availability-enabled="true">
            <ejb-name>ShoppingCartEJB</ejb-name>
            <checkpoint-at-end-of-method>
                <method>
                    <method-name>addToCart</method-name>
                </method>
            </checkpoint-at-end-of-method>
        </ejb>
        ...
    </enterprise-beans>
</sun-ejb-jar>
```

RMI-IIOP Load Balancing and Failover

This section describes using Sun Java System Application Server's high-availability features for remote EJB references and JNDI objects over RMI-IIOP.

- Overview
- Procedure for Application Client Container
- Procedure for Stand-Alone Client

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's view of the cluster. To do so, you must manually update the endpoints list on the client side.

Requirements

Sun Java System Application Server provides high availability of remote EJB references and `NameService` objects over RMI-IIOP. Before using these features, your environment must meet the following requirements:

- A cluster of at least two application server instances exists.
- 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.

  Load balancing is supported for the following RMI-IIOP clients accessing EJB components deployed on an Application Server instance.

  - Java applications executing in the Application Client Container (ACC). See "Procedure for Application Client Container" on page 62.
  - Java applications not running in the ACC. See "Procedure for Stand-Alone Client" on page 63

NOTE Application Server does not support RMI-IIOP load balancing and failover over secure sockets layer (SSL).

Algorithm

Sun Java System Application Server employs 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's primary endpoint becomes unavailable, further requests on that bean fail over to one of the alternate endpoints.

You can configure RMI-IIOP load balancing and failover to work with applications running in the ACC 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.

Procedure for 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 the *Sun Java System Application Server 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 the *Sun Java System Application Server Reference Manual*.
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 point 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 the `package-appclient` documentation in the *Sun Java System Application Server Reference Manual*.
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.

   You specify the IIOP listeners as endpoints to determine which IIOP listener receives the requests. The IIOP listeners are displayed 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.

      Make note of 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:*iiopport_of_instance1*,*machine2*:*iiopport_of_instance2*

   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 your application client in the following way:

   `appclient -client` *clientjar* `-name` *AppName*

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.

Procedure for Stand-Alone Client

To use RMI-IIOP load balancing and failover in a 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. Copy 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:3370
   0,host3.sun.com:33700 samples.rmiiiopclient.client.Standalone_Client
   ```

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.

Java Message Service Load Balancing and Failover

This section 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 the chapter "Using the Java Message Service" in the *Sun Java System Application Server Developer's Guide*. For more information on connectors (resource adapters), see the chapter "Developing Connectors" in the *Sun Java System Application Server Developer's Guide*.

For more information about the Sun Java System Message Queue, refer to the following documentation:

`http://docs.sun.com/app/docs/coll/MessageQueue_05q1`

For general information about the JMS API, see the JMS web page at:

`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 the *Sun Java System Application Server Administration Guide*.
- The `asadmin set` command. You can set the following attributes:

```
server.jms-service.init-timeout-in-seconds = 60
server.jms-service.type = LOCAL
server.jms-service.start-args =
server.jms-service.default-jms-host = default_JMS_host
server.jms-service.reconnect-interval-in-seconds = 60
server.jms-service.reconnect-attempts = 3
```

```
server.jms-service.reconnect-enabled = true
server.jms-service.addresslist-behavior = random
server.jms-service.addresslist-iterations = 3
server.jms-service.mq-scheme = mq
server.jms-service.mq-service = jms
```

You can also set these properties:

```
server.jms-service.property.instance-name = imqbroker
server.jms-service.property.instance-name-suffix =
server.jms-service.property.append-version = false
```

Use the `asadmin get` command to list all the Java Message Service attributes and properties. For details, type `man set` or `man get`, or see the *Sun Java System Application Server Reference Manual*.

You can override the Java Message Service configuration using JMS connection factory settings. For details, see the *Sun Java System Application Server 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 the *Sun Java System Application Server 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. The default host is 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 details, see the *Sun Java System Application Server Administration Guide*.
- Use the `asadmin create-jms-host` command. For details, see the *Sun Java System Application Server Reference Manual*.

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).
    - Priority—Java Message Service tries to connect to the MQ broker in the JMS Hosts list and uses another one only if the first broker is not available.
    - 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 the *Sun Java System Application Server 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 using 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, *Sun Java System Application Sever Deployment Planning Guide*.

Enabling MQ Clusters

To use MQ clusters with Application Server clusters, follow this procedure:

1. Create an Application Server cluster, if one does not already exist. For information on creating clusters, see the chapter "Configuring Clusters" in the *Sun Java System Application Server Administration Guide.*

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.

   You can create a JMS hosts with either the Admin Console or the `asadmin` command-line utility. For example, using `asadmin`:

   ```
   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 use Admin Console, navigate to the JMS Hosts node (Configurations > *config-name* > Java Message Service > JMS Hosts). Then:

   - Delete the default broker (default_JMS_host) by selecting the checkbox next to it, and then clicking Delete.
   - Click New to create each JMS host and 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, navigate to the JMS Hosts page (Configurations > *config-name* > Java Message Service > Physical Destinations). Then:

- Click New to create each JMS physical destination.
- 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, navigate to the JMS Connection Factories page (Resources > JMS Resources > Connection Factories). Then, to create each connection factory:

- Click New. The Create JMS Connection Factory page opens.
- For each connection factory, enter
  JNDI Name: for example "`jms/MyQcf`" and "`jms/MyQcf1`"
  Type: `javax.jms.QueueConnectionFactory`
- Select the cluster from the list of available targets at the bottom of the page, then click "Add."
- 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, navigate to the JMS Destination Resources page (Resources > JMS Resources > Connection Factories). Then, to create each destination resource:

- Click New. The Create JMS Destination Resource page opens.

- For each destination resource, enter
  JNDI Name: for example "jms/MyQueue" and "jms/MyQueue1"
  Type: javax.jms.Queue
- Select the cluster from the list of available targets at the bottom of the page, then click "Add."
- 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 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) used to store HTTP and stateful session bean (SFSB) session information. Without a session persistence mechanism, the HTTP or SFSB session state, including the passivated session state, is lost when a web or EJB container fails over to another. Use of the HADB for session persistence overcomes this situation.

This section contains procedures for setting up and configuring the HADB database for use by the Application Server.

About Highly Available Clusters

A highly available cluster in the Sun Java System Application Server Enterprise Edition integrates a state replication service with the clusters and load balancer.

HADB is a high availability database for storing session state. `HttpSession` objects and Stateful Session Bean state is stored in the HADB. This horizontally scalable state management service can be 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.

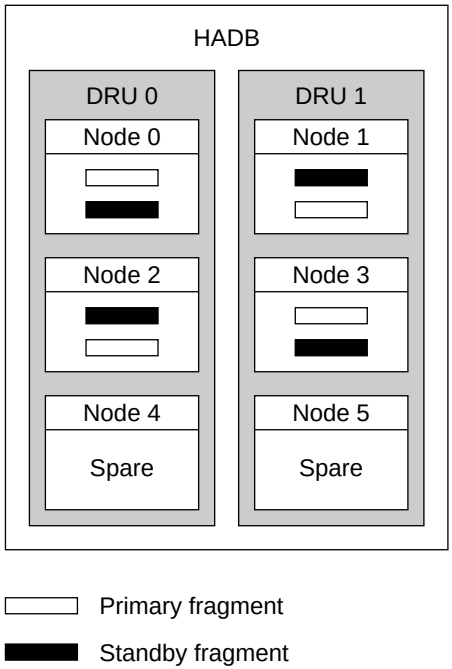
Keeping state management responsibilities separated from Application Server has significant benefits. Application Server instances spend their cycles performing as a scalable and high performance Java™ 2 Platform, Enterprise Edition (J2EE™ platform) containers delegating state replication to an external 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.

HADB Server Architecture

High availability means availability despite planned outages for upgrades or unplanned outages caused by hardware or software failures. The HADB is based on a simple data model and the Always-On technology. The 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



The 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. This process of fragmentation is based on a hash function. This hash function fragments and 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.

The HADB reclaims space when session data is deleted. The 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 details, see the *Sun Java System Application Server Enterprise Edition 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. It is used for storing and updating session data. Each node must have 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 the *Sun Java System Application Server System Deployment 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
- Specific Changes

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 multi-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`
  - `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.
- Changes made to `hadbm startnode` and `hadbm restartnode`.
    - Modified option:

      `startlevel` has a new value, `clear`.
- Changes made to `hadbm addnodes`.
    - New options: `set`, `historypath`, `devicepath`
    - Option removed: `inetdsetupdir`
- Changes made to `hadbm get` and `hadbm set`.
    - New attributes:
        - `historypath` (heterogeneous path for history files)
        - `packagename`
    - Removed attributes:
        - `managementProtocol`
        - `TotalDeviceSizePerNode`

- `installpath`
- `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
- Configuring Shared Memory and Semaphores
- Configuring Network Redundancy
- Synchronizing System Clocks
- File System Support

After performing these tasks, see Chapter 3, "Administering High Availability Database," for information on configuring and managing HADB.

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

Prerequisites

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

- 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:

- Set up network multipathing, supported only on Solaris 9.
- Configure a double network.

Setting Up Network Multipathing

Before setting up network multipathing, refer to the Administering Network Multipathing section of the *IP Network Multipathing Administration Guide* at http://docs.sun.com/doc/816-5249.

If the HADB host machines already use IP multipathing, configure them as follows:

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`.

   For example, if Host 1 and Host 2 have two physical network interfaces each, then on each host, these two interfaces are set up as a multipath group. Running `ifconfig -a` yields the following:

Host 1:

```
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
```

Host 2:

```
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 the ones listed as `bge0` and `bge1`. The ones listed as `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, the IP address 129.159.115.10 from host 1 and 129.159.115.20 from host 2 are selected for the use of HADB. To create a database with one database node per host, use the `--host` argument to `hadbm create`. 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 following argument:

```
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 Host 1 and Host 2 with separate parameters to specify which interface on the machines the agents should use:

```
Host 1: ma.server.mainternal.interfaces=129.159.115.10
Host 2: ma.server.mainternal.interfaces=129.159.115.20
```

For information on the `ma.server.mainternal.interfaces` variable, see Table 3-2 on page 99.

Configuring Double Networks

To allow HADB to tolerate single network failures, use IP multipathing if the operating system (for example, Solaris) supports it.

If your operating system is not configured for IP multipathing, and the 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.

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

When creating an HADB database, specify two IP addresses or host names for each node: one for each NIC IP address, using the --hosts option. 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 network addresses for node `0` are `10.10.116.61` and `10.10.124.61`, and the network addresses for node `1` are `10.10.116.62` and `10.10.124.62`. The addresses `10.10.116.61` and `10.10.116.62` are on the same subnet, and the addresses `10.10.124.61` and `10.10.124.62` are on the same subnet. 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
```

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

For the example above, 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.

Procedure for Solaris

Follow this procedure to configure shared memory and semaphores:

1. Log in as root.
2. Set the `shmmax` value 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 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
   ```

   `shmsys:shminfo_shmseg` is obsolete in Solaris 9 and later.

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

   NOTE The `shmsys:shminfo_shmmax` value is specified using the hexadecimal value for the memory size. To determine your host's memory, use this command:

   ```
   prtconf | grep Memory
   ```

3. 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. 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 contain the above mentioned entries, add these entries 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.

Procedure for Linux

Follow this procedure to configure shared memory and semaphores:

1. Log in as root.
2. Edit the file `/etc/sysctl.conf`
3. 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`

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. Also, HADB includes time stamps in events it logs 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, use:

- `xntpd` (network time protocol daemon) on Solaris
- `ntpd` on Linux

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

HADB has some restrictions 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 the *Sun Java System Application Server Performance Tuning 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`.

Setting Root Privileges for Node Supervisor Processes

The node supervisor processes (NSUP) ensure the availability of the HADB by exchanging “I’m alive” messages with each other. The NSUP executable files must have root privileges so that they can respond with real-time priority, as quickly as possible.

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

If NSUP executables do not have the proper privilege, 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.

Procedure

To avoid resource starvation, follow this procedure:

1. Log in with root privileges.

2. `cd HADB_install_dir/lib/server`

   where *HADB_install_dir* is the HADB installation directory.

3. The NSUP executable file is `clu_nsup_srv`.

   a. Set the file's `suid` bit.

      ```
      # chown root clu_nsup_srv
      ```

   b. set the file's ownership to root.

      ```
      # chmod u+s clu_nsup_srv
      ```

This starts the `clu_nsup_srv` process as the user root, and enables the process to give itself realtime priority. The `clu_nsup_srv` process does not consume significant CPU resources, has a small footprint, and running it with real-time priority does not affect performance.

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.

Restrictions

If NSUP cannot set the real-time priority `errno` is set to EPERM on Solaris and Linux. 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)

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

Prerequisites

Before configuring HADB, do the following:

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 the *Application Server 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 Administration Guide*.

3. Install and configure your web server software.

   For more information, see "Configuring Web Servers for HTTP Load Balancing" on page 24.

4. Setup and configure load balancing.

   For more information, see "Configuring the Load Balancer" on page 32.

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 97.

Procedure for Java Enterprise System

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

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.

Procedure for Standalone Application Server

This section describes how to start the management agent as a service when using the standalone Application Server.

To start the management agent as a service:

1. In a shell, change your current directory to *`HADB_install_dir`*`/lib`.
2. Edit the shell script `ma-initd` and 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 98.
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
```

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 *Sun Java System Application Server Administration Guide.*

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 the *Sun Java System Application Server System Reference Manual.*

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:

Procedure

To upgrade your HADB Installation to a newer version, follow these steps:

1. Install new version of HADB.
2. Unregister your existing HADB installationas described in "Unregistering HADB Packages" on page 94.
3. Register the new HADB version, as described in "Registering HADB Packages" on page 93.

   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 with 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 95.

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 Table 3-3 and Table 3-4 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 Table 2-1 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 Table 3-3 and Table 3-4 for a description of other command options.

For example:

Unregistering a 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
- Using the hadbm Management Command
- Configuring HADB
- Managing the HADB
- Expanding the HADB
- Monitoring HADB
- Maintaining HADB Machines

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
- Customizing Management Agent Configuration

- Starting the Management Agent

Management Agent Command Syntax

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

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

Where:

- *common-options* is one or more of the common options described in Table 3-1.
- *config-file* is the full path to the management agent configuration file. For more information, see "Customizing Management Agent Configuration" on page 98.

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 Table 3-2 on page 99. 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

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, follow this procedure:

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

The management agent configuration file is installed to the following location:

- Java Enterprise System. All the entries in the 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.
  - Solaris and Linux: `/etc/opt/SUNWhadb/mgt.cfg`.
- Standalone installer
  - Solaris and Linux: *`HADB_install_dir`*`/bin/ma.cfg`.

Table 3-2 describes the settings in the configuration file.

Table 3-2 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`
ma.server.type	Client protocol. Only JMXMP is supported.	jmxp
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

Table 3-2 Configuration File Settings

Setting Name	Description	Default
ma.server.dbdevicepath	Path to store HADB device information.	Solaris and Linux: `/var/opt/SUNWhadb/4`
ma.server.dbhistorypath	Path to store HADB history files.	Solaris and Linux: `/var/opt/SUNWhadb`
ma.server.dbconfigpath	Path to store node configuration data.	Solaris and Linux: `/var/opt/SUNWhadb/dbdef`
repository.dr.path	Path to domain repository files.	Solaris and Linux: `/var/opt/SUNWhadb/repository`

Starting the Management Agent

You can start the management agent two ways:

- As a service. In a production environment, start it as a service to ensure its availability. See "Starting the Management Agent as a Service."
- As a regular process (in console mode), for testing or development. See "Starting the Management Agent in Console Mode."

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

For a production deployment, start the management agent as a service to ensure its availability in case the `ma` process fails or the operating system reboots.

Procedure for Java Enterprise System

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

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

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.

The following procedure ensures 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 start the management agent as a service:

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.

Procedure for Standalone Application Server

This section describes how to start the management agent as a service when using the standalone Application Server.

To start the management agent as a service:

1. In a shell, change your current directory to *`HADB_install_dir`*`/lib`.
2. Edit the shell script `ma-initd` and 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 98.
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
   ```

   This procedure ensures the management agent starts only when the system enters runlevel 3 on Solaris or runlevel 5 on RedHat Linux. For a brief discussion of runlevels, see "Procedure for Java Enterprise System" on page 100.

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.

Java Enterprise System

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.

Standalone Application Server

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`.

For example, 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 manpage 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 109 and “Creating a Management Domain” on page 108.

Table 3-3 summarizes the hadbm security command line options.

Table 3-3 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. Table 3-4 describes the hadbm general command options.

Table 3-4 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.

Table 3-4 hadbm General Options *(Continued)*

Option (Short Form)	Description
--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. Table 3-5 describes environment variables that correspond to `hadbm` command options.

Table 3-5 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
--dbpassword	-p	none	$HADBM_DBPASSWORD
--dbpasswordfile	-P	none	$HADBM_DBPASSWORDFILE
--devicepath	-d	Solaris and Linux: `/var/opt/SUNWhadb`	$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`	$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

Table 3-5 HADB Options and Environment Variables *(Continued)*

Long Form	Short Form	Default	Environment Variable
--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
- Creating a Database
- Viewing and Modifying Configuration Attributes
- Configuring the JDBC Connection Pool

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 Table 3-4 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 109.

Example of 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.
```

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:

1. Create the management domain. See "Creating a Management Domain" on page 108 for more information.
2. Register the HADB package. See "Registering HADB Packages" on page 93 for more information.

If you have not performed the first two steps when you run `hadbm create`, it implicitly creates a domain and registers the HADB package. 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. 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.

Therefore, it is best to create the database only after creating the domain and registering the HADB package.

The command syntax is:

```
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 ]
[--adminpassword=password |--adminpasswordfile=file | --no-adminauthentication]
--hosts=host list [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 3-6 describes the special `hadbm create` command options. See Table 3-4 for a description of additional command options.

Table 3-6 `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 113.	Solaris and Linux: `/var/opt/SUNWhadb` Default is specified by `ma.server.dbdevicepath` in management agent configuration file. For more details, see "Configuration File Settings" on page 99.

Table 3-6 `hadbm create` Options *(Continued)*

Option (Short Form)	Description	Default
--devicesize=*size* -z	Device size for each node. For more information, see "Specifying Device Size" on page 112. Increase the device size as described in "Adding Storage Space to Existing Nodes" on page 133.	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 148.	Default is specified by `ma.server.dbhistorypath` in management agent configuration file. For more details, see "Configuration File Settings" on page 99. Solaris and Linux: `/var/opt/SUNWhadb`
--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. Host names See "Specifying Hosts" on page 112 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`

Table 3-6 `hadbm create` Options *(Continued)*

Option (Short Form)	Description	Default
--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 148.	None

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 81.

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 148.

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.

Example of Creating a Database

The following command is an example of creating a database:

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

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 100 and "Creating a Management Domain" on page 108.
- 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 80.
- Sun Java System Application Server and HADB port assignments must not conflict with other port assignments on the same machine. Default and recommended port assignments are as follows:
  - Sun Java System Message Queue: 7676
  - IIOP: 3700
  - HTTP server: 80
  - Admin server: 4848
  - HADB nodes: Each node uses six consecutive ports. If the default portbase `15200` is used, 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 (for a list, see "Configuration Attributes" on page 117), use the `hadbm get` command. 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 Table 3-7.

See Table 3-4 for a description of command options.

For example:

```
hadbm get JdbcUrl,NumberOfSessions
```

Setting the Values of Configuration Attributes

To set the values of configuration attributes (for a list, see "Configuration Attributes" on page 117), use the `hadbm set` command. The syntax is as follows:

```
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 Table 3-4 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 139. If the command does not execute successfully, restart the HADB as described in "Restarting the HADB" on page 129.

You cannot set the following attributes with `hadbm set`. Instead, set them when you create a database (see "Creating a Database" on page 109):

- 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 the 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

Table 3-7 lists the configuration attributes that you can modify with `hadbm set` and and retrieve with `hadbm get`.

Table 3-7 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
DataDeviceSize	Specifies the device size for the node. For information on the recommended `DataDeviceSize`, see "Specifying Device Size" on page 112. 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`	
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

Table 3-7 Configuration Attributes *(Continued)*

Attribute	Description	Default	Range
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
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`	
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 3-7 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` (see `man syslog` for details). 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 3-7 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	none alert error warning info
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 the 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*):

- Name of the connection pool is "*cluster-name*-hadb-pool".
- 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 134.

- Getting the JDBC URL
- Creating a Connection Pool
- Connection Pool Example
- Creating a JDBC Resource

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 the 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 the next section.

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 3-8 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 3-9 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.
serverList	JDBC URL of the HADB. To determine this value, see "Getting the JDBC URL" on page 120. You must change this value if you add nodes to the database. See "Adding Nodes" on page 134.
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`.

Connection Pool Example

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\\:p
ort,host\\:port,host\\:port,host\\:port:cacheDatabaseMetaData=false:eliminateRedund
antEndTransaction=true hadbpool
```

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

Creating a JDBC Resource

The following table summarizes JDBC resource settings required for the HADB.

Table 3-10 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 81.
Data Source Enabled	Checked/true

Managing the 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
- Managing Nodes
- Managing Databases
- Recovering from Session Data Corruption

Managing Domains

You can perform the following operations on an HADB domain:

- Creating a Domain: For information on the creating a domain, see "Creating a Management Domain" on page 108.
- Extending a Domain
- Deleting a Domain
- Listing Hosts in a Domain

- Removing Hosts from a Domain

See Table 3-3 on page 105 and Table 3-4 on page 105 for a description of command options. For more information on these commands, see the *Sun Java Application Server Reference Manual* or the corresponding man pages.

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.

Deleting a Domain

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

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

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
```

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]
```

Managing Nodes

You can perform the following operations on individual nodes:

- Starting a Node
- Stopping a Node
- Restarting a Node

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 the HADB" on page 130.

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.

Start level option

This 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 Table 3-4 for a description of other command options.

For example:

```
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`.

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

See Table 3-4 for a description of other command options.

For example:

```
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.

This 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 125 for more information.

See Table 3-4 for a description of other command options.

For example:

```
hadbm restartnode 1
```

Managing Databases

You can perform the following operations on HADB databases:

- Starting the HADB
- Stopping the HADB
- Restarting the HADB
- Listing Databases
- Clearing the HADB
- Removing a Database

Starting the HADB

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 Table 3-4 for a description of command options.

For example:

```
hadbm start
```

Stopping the HADB

When you stop and start the HADB in separate operations, data is unavailable while the HADB is stopped. To keep data available, you can restart the HADB as described in "Restarting the HADB" on page 129.

Stop the HADB to:

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

Before stopping the HADB, either stop dependent Application Server instances that are using the HADB 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 139.

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 Table 3-4 for a description of command options.

For example:

```
hadbm stop
```

Restarting the HADB

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

When you restart the HADB, data and database services remain available. When you stop and start the HADB in separate operations, data and database services are unavailable while the 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 Table 3-4 for a description of other command options.

For example:

```
hadbm restart
```

Listing Databases

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

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

See Table 3-4 for a description of command options.

Clearing the HADB

Clear the HADB when:

- `hadbm status` command reveals that the database is non-operational or if multiple nodes do not respond and in the waiting state for a long time. See "Getting the Status of HADB" on page 139.
- Recovering from session data corruption. See "Recovering from Session Data Corruption" on page 132.

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 Table 3-4 for a description of other options.

Table 3-11 `hadbm clear` Options

Option (Short Form)	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 from the HADB system, 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 the HADB" on page 128.

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 Table 3-4 for a description of command options.

For example, 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 Sun Java System Application Server system log (`server.log`) every time you try to save the session state.
- Error messages are written to the server log indicating 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 data has been corrupted, do the following:

1. Clear the session store.
2. If clearing the session store doesn't work or you continue to see errors in the server log, reinitialize the data space on all the nodes and clear the data in the database. See "Clearing the HADB" on page 130.
3. If clearing the database doesn't work, delete and then recreate the database. See "Removing a Database" on page 131 and "Creating a Database" on page 109.

Expanding the 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 the HADB without shutting down your Application Server cluster or database, in particular:

- Adding Storage Space to Existing Nodes

- Adding Machines
- Adding Nodes
- Refragmenting the Database
- Adding Nodes by Recreating the Database

Also see related information in "Maintaining HADB Machines" on page 146.

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 142.

If there is unused space on the disks on which the HADB nodes reside, or if you added or upgraded the disks with larger capacity, increase the data device size. For information on the recommended data device size, see "Specifying Device Size" on page 112.

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 140.

The command syntax is:

`hadbm set DataDeviceSize=`*size*

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

See Table 3-4 for a description of command options.

For example:

```
hadbm set DataDeviceSize=1024
```

Adding Machines

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

To integrate new machines into the 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 more details, see "Adding Nodes" on page 134.

Adding Nodes

Increase processing and storage capacity of the HADB system by creating new nodes and adding 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 detailed information, see the *Application Server 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 the HA Fault Tolerant or Fault Tolerant state. For more information about database states, see "Getting the Status of HADB" on page 139.

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.

For example:

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

The following table describes the special `hadbm addnodes` command options. See Table 3-4 for a description of other options.

Table 3-12 `hadbm addnodes` Options

Option (Short Form)	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 136. 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 137.	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
--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 113.	Solaris and Linux: *HADB_install_dir* `/device`
--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 81.	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 the HA Fault Tolerant or Fault Tolerant state. For more information about database states, see "Getting the Status of HADB" on page 139.

See Table 3-4 for a description of command options.

For example:

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

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

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 142.

If this command fails after multiple attempts, see "Adding Nodes by Recreating the Database" on page 137.

Adding Nodes by Recreating the Database

If online refragment fails persistently when new nodes are added, 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, as well as schema data.

To add nodes by recreating the database:

1. For each Application Server instance:
   a. Disable the Application Server instance in the load balancer.
   b. Disable session persistence.
   c. Restart the 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 the HADB" on page 128.
3. Delete the database as described in "Removing a Database" on page 131.
4. Recreate the database with the additional nodes as described in "Creating a Database" on page 109.
5. Reconfigure the JDBC connection pool as described in "Configuring the JDBC Connection Pool" on page 120.
6. Reload the session persistence store.
7. Perform the following tasks 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
- Getting Device Information
- Getting Runtime Resource Information

These sections briefly describe the `hadbm status`, `hadbm deviceinfo`, and `hadbm resourceinfo` commands. For details about interpreting HADB information, see the *Sun Java System Application Server 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. See "Node Status" on page 140. See Table 3-4 for a description of other command options.

For example:

```
hadbm status --nodes
```

Database States

A database's *state* summarizes its current condition. Table 3-13 describes the possible database states.

Table 3-13 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 the HADB" on page 130.
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 141.
- State of the node. For a list of states and their meanings, see "States of a Node" on page 141.
- Number of the corresponding mirror node.

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

- Roles of a Node
- States of a Node

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 126 and "Starting a Node" on page 125. If multiple nodes are in this state, clear the database as described in "Clearing the HADB" on page 130.
- **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).
- Free size (in MB).
- Usage (in percent).

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 Table 3-4 for a description of other command options.

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*.

For example, 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 the *Sun Java System Application Sever Deployment Planning Guide* and the *Sun Java System Application Server 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.

Table 3-14 describes the `hadbm resourceinfo` special command options. See Table 3-4 for a description of other command options.

Table 3-14 `hadbm resourceinfo` Command Options

Option (Short Form)	Description
--databuf	Display data buffer pool information.
-d	See "Data Buffer Pool Information" below for more information.
--locks	Display lock information.
-l	See "Lock Information" below for more information.
--logbuf	Display log buffer information.
-b	See "Log Buffer Information" below for more information.
--nilogbuf	Display node internal log buffer information.
-n	See "Node Internal Log Buffer Information" 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.

For example:

```
NodeNO Avail  Free  Access  Misses Copy-on-Write

0      256    128   100000  50000  1000
1      256    128   110000  45000  950
```

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.

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.

For example:

```
NodeNO Avail  Free  Waits

0      50000  20000 10
1      50000  20000 0
```

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 148.

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

For example:

```
NodeNO Avail  Free

0       16      2
1       16      3
```

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.
- Received by the mirror node.

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

For example:

```
NodeNO Avail  Free

0       16      2
1       16      3
```

Maintaining HADB Machines

The HADB achieves fault tolerance by replicating data on mirror nodes. Mirror nodes should be placed on separate DRUs in a production environment as described in *Sun Java System Application Sever 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, the 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 the HADB and recreate its session store, which erases all its data.

Maintaining HADB on a Single Machine

To perform planned or unplanned maintenance on a single machine without interrupting HADB service:

1. Perform the maintenance procedure and get the machine up and running.
2. Ensure that `ma` is running: If `ma` runs 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 100.
3. Start all nodes on the machine.

   See "Starting a Node" on page 125.
4. Check whether the nodes are active and running. See "Getting the Status of HADB" on page 139.

To perform planned maintenance, such as hardware or software upgrade, on all HADB machines without interrupting HADB service:

1. For each spare machine in the first DRU, repeat the single machine procedure as described in "Maintaining HADB on a Single Machine" on page 147, one by one, for each machine.
2. For each active machine in the first DRU, repeat the single machine procedure as described in "Maintaining HADB on a Single Machine" on page 147, one by one, for each machine.
3. Repeat Step 1 on page 147 and Step 2 on page 147 for the second DRU.

To perform planned maintenance with HADB service interruption on all HADB machines, or when the entire HADB is on a single machine:

1. Stop the HADB. See "Stopping the HADB" on page 128.
2. Perform the maintenance procedure and get all the machines up and running.

3. Ensure `ma` is running.
4. Start the HADB. See "Starting the HADB" on page 128. The data stored in the database before the stop is available again.

To perform unplanned maintenance in the event of a failure, first check the database status. See "Getting the Status of HADB" on page 139.

- If the database state is Operational or better, this means 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, this means 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 the HADB and recreate the session store. See "Clearing the HADB" on page 130. This interrupts HADB service.

Clearing and Archiving History Files

HADB history files record all database operations and error messages. HADB appends on 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` (short form `-o`) to specify the directory in which to store the old history files. This directory must have appropriate write permissions. See Table 3-4 for a description of other command options

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 109.

History File Format

Each message in the history file contains the following information:

- The abbreviated 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, obtain help from Sun customer support.

Index

A

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W