Installation and Tuning Guide

Sun™ ONE Directory Server

Version 5.2

816-6697-10
June 2003

SunTone™ CERTIFIED

Contents

About This Guide

Sun™ ONE Directory Server 5.2 is a powerful and scalable distributed directory service based on the industry-standard Lightweight Directory Access Protocol (LDAP). Sun ONE Directory Server software is part of the Sun Open Net Environment (Sun ONE), Sun's standards-based software vision, architecture, platform, and expertise for building and deploying Services On Demand.

Sun ONE Directory Server is the cornerstone for building a centralized and distributed data repository that can be used in your intranet, over your extranet with your trading partners, or over the public Internet to reach your customers.

Purpose of This Guide

This guide demonstrates how to install Directory Server for use in a production environment. Preparing Directory Server for high performance in a production environment often involves considerable configuration and tuning.

If you are installing Directory Server for evaluation purposes only and not for use in a production environment, you may choose to read only Chapter 1, "Installing Sun ONE Directory Server."

Prerequisites

Before installing Directory Server for use in a production environment, make sure your deployment objectives are clear. Refer to the *Sun ONE Directory Server Deployment Guide* for more information.

Typographical Conventions

This section explains the typographical conventions used in this book.

`Monospaced font` - This typeface is used for literal text, such as the names of attributes and object classes when they appear in text. It is also used for URLs, filenames, and examples.

Italic font - This typeface is used for emphasis, for new terms, and for text that you must substitute for actual values, such as placeholders in path names.

The greater-than symbol (>) is used as a separator when naming an item in a menu or sub-menu. For example, Object > New > User means that you should select the User item in the New sub-menu of the Object menu.

NOTE Notes, Cautions, and Tips highlight important conditions or limitations. Be sure to read this information before continuing.

Default Paths and Filenames

All path and filename examples in the Sun ONE Directory Server product documentation are one of the following two forms:

- *ServerRoot*/... - The *ServerRoot* is the location of the Sun ONE Directory Server product. This path contains the shared binary files of Directory Server, Sun ONE Administration Server, and command line tools.

  The actual *ServerRoot* path depends on your platform, your installation, and your configuration. The default path depends on the product platform and packaging as shown in Table 1.

- *ServerRoot*/`slapd-`*serverID*/... - The *serverID* is the name of the Directory Server instance that you defined during installation or configuration. This path contains database and configuration files that are specific to the given instance.

NOTE Paths specified in this manual use the forward slash format of UNIX and commands are specified without file extensions. If you are using a Windows version of Sun ONE Directory Server, use the equivalent backslash format. Executable files on Windows systems generally have the same names with the `.exe` or `.bat` extension.

Table 1 Default *ServerRoot* Paths

Product Version	*ServerRoot* **Path**
Solaris Packages[1]	`/var/mps/serverroot` - After configuration, this directory contains links to the following locations: • `/etc/ds/v5.2` (static configuration files) • `/usr/admserv/mps/admin` (Sun ONE Administration Server binaries) • `/usr/admserv/mps/console` (Server Console binaries) • `/usr/ds/v5.2` (Directory Server binaries)
Compressed Archive Installation on Solaris and Other UNIX Systems	`/var/Sun/mps`
Zip Installation on Windows Systems	`C:\Program Files\Sun\MPS`

1. If you are working on the Solaris Operating Environment and are unsure which version of the Sun ONE Directory Server software is installed, check for the existence a key package such as `SUNWdsvu` using the `pkginfo` command. For example: `pkginfo | grep SUNWdsvu`.

Directory Server instances are located under *ServerRoot*`/slapd-`*serverID*`/`, where *serverID* represents the server identifier given to the instance on creation. For example, if you gave the name `dirserv` to your Directory Server, then the actual path would appear as shown in Table 2. If you have created a Directory Server instance in a different location, adapt the path accordingly.

Table 2 Default Example `dirserv` Instance Locations

Product Version	**Instance Location**
Solaris Packages	`/var/mps/serverroot/slapd-dirserv`
Compressed Archive Installation on Solaris and Other UNIX Systems	`/usr/Sun/mps/slapd-dirserv`
Zip Installation on Windows Systems	`C:\Program Files\Sun\MPS\slapd-dirserv`

Downloading Directory Server Tools

Some supported platforms provide native tools for accessing Directory Server. More tools for testing and maintaining LDAP directory servers, download the Sun ONE Directory Server Resource Kit (DSRK). This software is available at the following location:

```
http://wwws.sun.com/software/download/
```

Installation instructions and reference documentation for the DSRK tools is available in the *Sun ONE Directory Server Resource Kit Tools Reference*.

For developing directory client applications, you may also download the iPlanet Directory SDK for C and the iPlanet Directory SDK for Java from the same location.

Additionally, Java Naming and Directory Interface (JNDI) technology supports accessing the Directory Server using LDAP and DSML v2 from Java applications. Information about JNDI is available from:

```
http://java.sun.com/products/jndi/
```

The JNDI Tutorial contains detailed descriptions and examples of how to use JNDI. It is available at:

```
http://java.sun.com/products/jndi/tutorial/
```

Suggested Reading

Sun ONE Directory Server product documentation includes the following documents delivered in both HTML and PDF:

- *Sun ONE Directory Server Getting Started Guide* - Provides a quick look at many key features of Directory Server 5.2.
- *Sun ONE Directory Server Deployment Guide* - Explains how to plan directory topology, data structure, security, and monitoring, and discusses example deployments.
- *Sun ONE Directory Server Installation and Tuning Guide* - Covers installation and upgrade procedures, and provides tips for optimizing Directory Server performance.
- *Sun ONE Directory Server Administration Guide* - Gives the procedures for using the console and command-line to manage your directory contents and configure every feature of Directory Server.

- *Sun ONE Directory Server Reference Manual* - Details the Directory Server configuration parameters, commands, files, error messages, and schema.
- *Sun ONE Directory Server Plug-In API Programming Guide* - Demonstrates how to develop Directory Server plug-ins.
- *Sun ONE Directory Server Plug-In API Reference* - Details the data structures and functions of the Directory Server plug-in API.
- *Sun ONE Server Console Server Management Guide* - Discusses how to manage servers using the Sun ONE Administration Server and Java based console.
- *Sun ONE Directory Server Resource Kit Tools Reference* - Covers installation and features of the Sun ONE Directory Server Resource Kit, including many useful tools.

Other useful information can be found on the following Web sites:

- Product documentation online: `http://docs.sun.com/coll/S1_DirectoryServer_52`
- Sun software: `http://wwws.sun.com/software/`
- Sun ONE Services: `http://www.sun.com/service/sunps/sunone/`
- Sun Support Services: `http://www.sun.com/service/support/`
- Sun ONE for Developers: `http://sunonedev.sun.com/`
- Training: `http://suned.sun.com/`

Part 1

Installation

Chapter 1

Installing Sun ONE Directory Server

This chapter is designed to guide you through initial Sun ONE Directory Server software installation and uninstallation. It contains the following sections:

- Before You Start
- Installation
- Uninstallation
- Troubleshooting

Before You Start

Before you install Directory Server for use in a production environment, ensure the system is minimally equipped and configured to run directory services. At minimum, familiarize yourself with the concepts discussed in *Sun ONE Directory Server Deployment Guide*.

NOTE To achieve optimal performance, also follow the tuning and configuration instructions provided throughout this guide.

Planning Your Directory Deployment

Perform the following steps, referring to the operating system documentation for tasks related to the underlying platform.

1. Plan the deployment of directory services.

   Refer to the *Sun ONE Directory Server Deployment Guide* for instructions.

2. If the deployment involves centralized administration of server configuration, users, and groups for multiple directory installations, determine configuration and user directory locations.

   The *configuration directory* or Configuration Directory Server (CDS) stores information about how Directory Server itself is configured. This directory is generally installed first, and every subsequent server registers with it. A single configuration directory provides for centralized administration of all servers.

   The *user directory* stores entries for users and groups who access directory services. The user directory is generally unique to the network domain, and other servers access it for user and group information. A single user directory provides for centralized administration of users and groups.

   For small deployments, it is possible to install configuration, user, and other directories on the same directory instance. For larger deployments, consider placing the configuration and user directories on separate servers.

   Refer to the *Sun ONE Server Console Server Management Guide* for details on appropriate location of configuration, user, and group data.

3. Ensure the host system runs a supported platform on a supported architecture, as summarized in Table 1-1.

Table 1-1 Supported Platforms and Architectures

Platform	**Architectures**
Sun Solaris™ Operating Environment 9	SPARC® processors, 32 and 64-bit mode
	Supported x86 platforms
Sun Solaris Operating Environment 8	UltraSPARC processors, 32 and 64-bit mode
Sun Linux 5.0	Sun LX50 servers
Hewlett Packard HP-UX 11i	PA-RISC 2.0 processors, 32 and 64-bit mode
IBM AIX 5.1	PowerPC processors
Microsoft Windows 2000 Server, SP 3	Pentium II or later IA-32 processors
Microsoft Windows 2000 Advanced Server, SP 3	
Red Hat Linux 7.2	Pentium II or later IA-32 processors

4. Ensure the host system meets at least minimum disk space and memory requirements, as summarized briefly in Table 4-1 on page 80.
5. Restrict physical access to the host system.
6. Ensure the host system uses a static IP address.
7. If the Directory Server instance is not itself providing a naming service for the network or if the deployment involves remote administration of Directory Server, ensure a naming service and the domain name for the host are properly configured.

Obtaining Directory Server Software

After performing the procedure outlined in “Planning Your Directory Deployment,” on page 17, complete the following steps.

1. Ensure an unzip utility is installed so you can unpack the software.
2. Download the software. At the time of this writing, you can download from:

   `http://wwws.sun.com/software/download/`
3. Unpack the software into a directory other than the directory where you intend to install Directory Server.

Installation

Which Directory Server installation steps you follow depends on your specific deployment requirements. With these specific deployment requirements in mind, proceed according to the appropriate sections:

- Determining What to Install
- Determining How to Install
- Preparing Installation Information
- Installing on Solaris Systems
- Installing on Other UNIX Systems
- Installing on Windows Systems

Determining What to Install

You have a number of alternatives to evaluate before you decide which software to install. Consider these questions:

- Do you need large cache capabilities for a high-volume deployment?

  If so, consider using a platform on which Directory Server can run as a 64-bit process, and install the 64-bit version.

  If your Directory Server deployment is relatively small, with a database size of less than 500 MB, consider installing only 32-bit support, even on platforms that support 64-bit versions.

- Do you plan to administer Directory Server through the graphical user interface?

  If so, install Sun ONE Server Console and Sun ONE Administration Server.

  If you intend to administer Directory Server through the command-line interface only, then you may choose not to install the Console and Administration Server.

  If you intend to use the system for remote management through the graphical user interface, you may choose to install only the Console and Administration Server.

- Do you intend to deploy Directory Server on Sun Cluster software?

  If so, refer to Appendix C, "Installing Sun Cluster HA for Directory Server," for instructions.

Determining How to Install

You also have alternatives to evaluate before you decide which packaging best fits your deployment, and whether you intend to install interactively. Consider these questions:

- Do you want tighter integration with Solaris system administration processes? Do you want to share components among multiple Sun ONE servers on the same system?

  If so, consider installing using Solaris Packages.

- Do you want to install without first becoming super user? Do you want to install multiple independent sets of Directory Server binaries on the same system?

  If so, consider installing from a compressed archive, even on Solaris systems.

- Do you want to install quickly to evaluate Directory Server? Is this your first time installing this version of Directory Server?

  If so, consider installing interactively.

- Do you want to script installation? Do you want to install many systems with similar configurations?

  If so, consider using the silent installation process.

Preparing Installation Information

Preparing information in advance can help you complete the installation process quickly. Before performing interactive installations, consider creating a work sheet to hold the installation information, as summarized for a typical installation in Table 1-2.

Table 1-2 Basic Information Required During Typical Installation

Description	**Examples**	**Your Answers...**
Administration domain	`example.com`	
Administration Server port number	`5201`	
Directory Administrator ID	`admin`	
Directory Administrator password	`$3kReT4wD`	
Directory Manager DN[1] (super user for the directory)	`cn=Directory Manager`	
Directory Manager password (at least 8 characters)	`#$8Yk$-%&`	
Directory Server port number (1-65535, inclusive)[2]	`389` (default LDAP) `636` (default LDAP/SSL)	
Fully qualified host distinguished name	`dirserv.example.com`	

(Optional) Configuration directory host, port, bind ID and password if using an existing configuration directory	`config.example.com`
	`389`
	`admin`
	`$3kReT4wD`	

Table 1-2 Basic Information Required During Typical Installation *(Continued)*

Description	Examples	Your Answers...

(Optional) User directory host, port, bind DN, password, and suffix if using an existing user directory	`usergroup.example.com`
	`389`
	`cn=Directory Manager`
	`#$8Yk$-%&`
	`dc=example, dc=com`	

Server ID (No periods or spaces allowed)	`dirserv`	
Server suffix (At least one to hold directory content)	`dc=example,dc=com`	

ServerRoot (software installation directory; refer to "Default Paths and Filenames," on page 10 for more information)
 	
Do not install on top of an existing earlier version.
 	
Do not install Sun ONE Web Server in the same *ServerRoot* as Directory Server.
 	
(UNIX platforms) No spaces allowed.
	`/var/mps/serverroot`
	
	`/var/Sun/mps`
	
	`C:\Program Files\Sun\MPS`	

(UNIX platforms) Server group ID[3] Use the name, rather than the group ID number.	`noaccess`	
(UNIX platforms) Server user ID Use the name, rather than the user ID number.	`diruser`	
(Windows) `Administrator` password (Optional, other platforms) super user password	Ask your system administrator.	

1. All DNs must be entered in UTF-8 encoding; refer to RFC 2253. Older encodings such as ISO-8859-1 are not supported.
2. The Internet Assigned Numbers Authority assigns port numbers less than 1024. Install as super user to use a port less than 1024.
3. You create the appropriate UNIX user and group as described in the installation procedures.

When providing information for Directory Administrator and Directory Manager accounts, recall that Directory Administrator access rights may be managed using Directory Server access control mechanisms. Recall also that Directory Server access control does not apply for the Directory Manager account.

Silent installation configuration files contain similar information.

Installing on Solaris Systems

How you install Directory Server software depends on which packaging you decide to use, and on whether you want to interact with the install program. Proceed according to instructions in the appropriate sections:

- Preparing For Installation From Solaris Packages
- Performing Interactive Installation Using Solaris Packages
- Performing Silent Installation Using Solaris Packages

- Preparing For Installation From a Compressed Archive
- Performing Interactive Installation From a Compressed Archive
- Performing Silent Installation From a Compressed Archive

- Completing the Installation Process

When installing Directory Server in a Sun Cluster system, follow the instructions in Appendix C, "Installing Sun Cluster HA for Directory Server."

Preparing For Installation From Solaris Packages

1. (Optional) Create a user and group account for Directory Server.

   Directory Server runs as the user and group you specify during installation. Set permissions that prevent unauthorized access to the directory and to other resources on the system. Refer to "(UNIX Platforms) Users and Groups," on page 99 for more information.

2. (Optional) Allow access to the display using the `xhost`(1) command.

   When you set the `DISPLAY` environment variable appropriately and perform installation as a user having access to the display, the installation program displays the graphical user interface by default.

   If the installation program cannot display the graphical user interface, it starts installation in command-line mode.

3. Before installing using a locale other than US English, set the `LANG` environment variable to `C`.

4. Ensure the required packages listed in Table 1-3 are installed, in addition to all Solaris packages installed by default with a basic system.

Table 1-3 Prerequisite Solaris Packages

Package	Description	Required for 32-Bit Directory Server	Required for 64-Bit Directory Server
`SUNWj3rt`[1]	J2SDK 1.4 runtime environment	Yes	Yes
`SUNWzlib`	The Zip compression library	Yes	Yes
`SUNWzlibx`	The Zip compression library (64-bit)	No	Yes

1. It is strongly recommended that you use a Java Runtime Environment version 1.4.1 or later.

Performing Interactive Installation Using Solaris Packages

Perform the steps in the following procedures.

Installing Solaris Packages

You install Solaris packages using the `pkgadd`(1M) utility. Use `pkginfo`(1) to determine which packages are already installed, when performing an upgrade for example. When installing packages on multiple hosts, you may define default installation actions through the installation defaults file described in `admin`(4). In any case, all packages must share the same *basedir*.

Refer to the Solaris Operating Environment system administration documentation for further information on handling software packages.

1. Consider the full list of packages listed in Table 1-4 or Table 1-5.

Table 1-4 Solaris Packages Provided (SPARC Platforms)

Package	Description
`SUNWasha`	Sun ONE Administration Server Component for Sun Cluster
`SUNWasvc`	Sun ONE Administration Console
`SUNWasvcp`	Sun ONE Administration Server Console Plug-In
`SUNWasvr`	Sun ONE Administration Server (Root)
`SUNWasvu`	Sun ONE Administration Server (Usr)
`SUNWdsha`	Sun ONE Directory Server Component for Sun Cluster
`SUNWdsvcp`	Sun ONE Directory Server Console Plug-In
`SUNWdsvh`	Sun ONE Directory Server Heap Allocator (Solaris 8 systems only)
`SUNWdsvhx`	Sun ONE Directory Server Heap Allocator (64-bit, Solaris 8 systems only)

Table 1-4 Solaris Packages Provided (SPARC Platforms) *(Continued)*

Package	Description
SUNWdsvpl	Sun ONE Directory Server PerLDAP modules
SUNWdsvr	Sun ONE Directory Server (Root)
SUNWdsvu	Sun ONE Directory Server (Usr)
SUNWdsvx	Sun ONE Directory Server (64-bit)
SUNWicu	International Components for Unicode User Files
SUNWicux	International Components for Unicode User Files (64-bit)
SUNWjss	Network Security Services for Java (JSS)
SUNWldk	LDAP C SDK
SUNWldkx	LDAP C SDK (64-bit)
SUNWpr	Netscape Portable Runtime Interface
SUNWprx	Netscape Portable Runtime Interface (64-bit)
SUNWsasl	Simple Authentication and Security Layer
SUNWsaslx	Simple Authentication and Security Layer (64-bit)
SUNWtls	Network Security Services
SUNWtlsx	Network Security Services (64-bit)

Table 1-5 Solaris Packages Provided (x86 Platforms)

Package	Description
SUNWasvc	Sun ONE Administration Console
SUNWasvcp	Sun ONE Administration Server Console Plug-In
SUNWasvr	Sun ONE Administration Server (Root)
SUNWasvu	Sun ONE Administration Server (Usr)
SUNWdsvcp	Sun ONE Directory Server Console Plug-In
SUNWdsvpl	Sun ONE Directory Server PerLDAP modules
SUNWdsvr	Sun ONE Directory Server (Root)
SUNWdsvu	Sun ONE Directory Server (Usr)
SUNWicu	International Components for Unicode User Files
SUNWjss	Network Security Services for Java (JSS)

Table 1-5 Solaris Packages Provided (x86 Platforms) *(Continued)*

Package	Description
`SUNWldk`	LDAP C SDK
`SUNWpr`	Netscape Portable Runtime Interface
`SUNWsasl`	Simple Authentication and Security Layer
`SUNWtls`	Network Security Services

It is recommended that you use a writable *basedir* such as `/var` when installing all packages. Notice when relocating packages that `SUNWasvr` and `SUNWdsvr` place startup and shutdown scripts in *basedir*`/etc`.

2. Use the hints in Table 1-6 to determine which packages to install.

Table 1-6 Which Packages to Install

Configuration	List of Packages to Install[1]
32-bit Directory Server, Administration Server, and Console	`SUNWascv SUNWasvcp SUNWasvr SUNWasvu SUNWdsvcp SUNWdsvh SUNWdsvpl SUNWdsvr SUNWdsvu SUNWicu SUNWjss SUNWldk SUNWpr SUNWsasl SUNWtls`
32-bit Directory Server only (no Console)	`SUNWasvu SUNWdsvh SUNWdsvpl SUNWdsvr SUNWdsvu SUNWicu SUNWjss SUNWldk SUNWpr SSUNWsasl SUNWtls`
64-bit Directory Server, 32-bit Administration Server, and Console	`SUNWascv SUNWasvcp SUNWasvr SUNWasvu SUNWdsvcp SUNWdsvh SUNWdsvhx SUNWdsvpl SUNWdsvr SUNWdsvu SUNWdsvx SUNWicu SUNWicux SUNWjss SUNWldk SUNWldkx SUNWpr SUNWprx SUNWsasl SUNWsaslx SUNWtls SUNWtlsx`
64-bit Directory Server only (no Console)	`SUNWasvu,SUNWdsvh SUNWdsvhx SUNWdsvpl SUNWdsvr SUNWdsvu SUNWdsvx SUNWicu SUNWicux SUNWjss SUNWldk SUNWldkx SUNWpr SUNWprx SUNWsasl SUNWsaslx SUNWtls SUNWtlsx`
Cluster node	Add `SUNWasha SUNWdsha`
Sun ONE Server Console and Administration Server only (no Directory Server, remote management only)	`SUNWasvc SUNWasvcp SUNWasvr SUNWasvu SUNWdsvcp SUNWjss SUNWldk SUNWpr SUNWsasl SUNWtls`

1. Packages `SUNWdsvh` (32-bit) and `SUNWdsvhx` (64-bit) are required by Directory Server only on Solaris 8 systems.

3. Verify that the packages you want are not yet installed.

   Do not reinstall packages that have already been installed on the system.

4. Become super user.

5. Use the `pkgadd`(1M) utility to transfer product packages to the system.

   Packages `SUNWicu`, and `SUNWicux` depend on the version of Solaris running on the system where you install Directory Server.

   Furthermore, refer to the subsequent section, “Installing Required Patches,” for more information about installing and patching component packages `SUNWpr`, `SUNWprx`, `SUNWsasl`, `SUNWsaslx`, `SUNWtls`, and `SUNWtlsx`.

6. After quitting `pkgadd`, verify that all required product packages are installed.

When upgrading from iPlanet Directory Server 5.1 installed using `IPLT*` Solaris packages, the 5.1 `/usr/sbin/directoryserver` command is renamed to `/usr/sbin/directoryserver.51bak`. You may manage the 5.1 version using the renamed command.

Installing Required Patches

Directory Server relies on packages `SUNWpr`, `SUNWprx`, `SUNWsasl`, `SUNWsaslx`, `SUNWtls`, and `SUNWtlsx` that have been updated to include recent fixes, and on recommended system patches.

1. Using `pkginfo`(1) with the `-x` option, determine which of these packages are installed on your system. Verify specifically that the appropriate package versions have been installed for your system, as shown in Table 1-7.

Table 1-7 Appropriate Versions and Patches For Components

System Version and Architecture	SUNWpr(x) Version	SUNWsasl(x) Version	SUNWtls(x) Version	Patches
Solaris 9 (SPARC platforms)	4.1.2 or later	2.01 or later	3.3.2 or later	114049, 115342
Solaris 9 (x86 platforms)	4.1.3 or later	2.01 or later	3.3.3 or later	114050, 115343
Solaris 8 (SPARC platforms)	4.1.2 or later	2.01 or later	3.3.2 or later	114045, 115328

2. Using `showrev`(1M) with the `-p` option, determine whether the appropriate patches listed in Table 1-7 have been applied for your platform.

3. Use the hints in Table 1-8 to determine whether to patch components.

Table 1-8 Whether to Patch Components

On your system...	Do this...
The packages are already installed, and the patches have been applied.	Proceed to Step 4.
The packages are already installed, but the patches have not been applied.	Apply the appropriate patches for your platform provided with Directory Server.
The packages are not yet installed.	Install the packages and appropriate patches provided with Directory Server.

4. Run the following command as super user:

   ```
   root# /usr/sbin/directoryserver idsktune -q > idsktune.out
   ```

   `idsktune` suggests changes you may make to the system. The subcommand itself makes no changes to the system.

5. Fix at least all `ERROR` conditions indicated.

   If you do not fix `ERROR` conditions, installation may fail. Notice that the `idsktune` subcommand reports as missing *all* patches recommended at the time of release and not installed on the system, even patches for packages not installed on the system.

   You may download patches from `http://sunsolve.sun.com/`.

   Refer to Chapter 5, “Tuning the Operating System” for more information.

Configuring Directory Server

1. Start the configuration program.

   To use the graphical user interface:

   ```
   root# /usr/sbin/directoryserver configure
   ```

   To use the command-line interface:

   ```
   root# /usr/sbin/directoryserver configure -nodisplay
   ```

   The first installation screen appears.

2. Follow the instructions on each screen using the work sheet you made when “Preparing Installation Information,” on page 21.

Configuring Administration Server

1. Start the configuration program.

   To use the graphical user interface:

   ```
   root# /usr/sbin/mpsadmserver configure
   ```

   To use the command-line interface:

   ```
   root# /usr/sbin/mpsadmserver configure -nodisplay
   ```

   The first installation screen appears.

2. Follow the instructions on each screen using the work sheet you made when "Preparing Installation Information," on page 21.

Proceed to "Completing the Installation Process," on page 33.

Performing Silent Installation Using Solaris Packages

Complete the steps in the following procedures.

Installing Solaris Packages

Follow the instructions in "Installing Solaris Packages," on page 24.

Installing Required Patches

Follow the instructions in "Installing Required Patches," on page 27.

Creating Specification Files

To perform full silent installation, you must first create two files containing installation specifications, one for Directory Server, one for Administration Server. For a Directory Server installation specification file template, refer to `/usr/ds/v5.2/setup/typical.ins`. For Administration Server, refer to `/usr/sadm/mps/admin/v5.2/setup/admin/typicalInstall.ins`.

NOTE Specification files may contain passwords in clear text. Protect such files with appropriate file permissions.

You may create a silent installation specification file either by editing a copy of the template file by hand, or by performing interactive configuration using the Directory Server and Administration Server configuration programs.

To create silent installation specification files for Directory Server and for Administration Server interactively, follow these steps:

1. Perform Directory Server configuration using the `-saveState` option.

   `root# /usr/sbin/directoryserver configure -saveState` *dirserv-file*

   to create the specification file, *dirserv-file*.

2. Perform Administration Server configuration using the `-saveState` option.

   `root# /usr/sbin/mpsadmserver configure -saveState` *admserv-file*

   to create the specification file, *admserv-file*.

3. Adjust the specification files, *dirserv-file* and *admserv-file*, before using them to install on other systems.

   Some silent installation specification file directives, such as `FullMachineName`, depend directly on the underlying host system and so cannot be generated generically.

Silent installation specification files contain a checksum string corresponding to the build version of the install program. To reuse a silent installation specification file with a different build or release of the install program, update the checksum string in lines beginning with `[STATE_BEGIN` and `[STATE_DONE`. The updated checksums are in `/usr/ds/v5.2/setup/typical.ins` for Directory Server and in `/usr/sadm/mps/admin/v5.2/setup/admin/typicalInstall.ins` for Administration Server. Code Example 1-1 shows a sample checksum.

Code Example 1-1 Silent Installation Checksum Line

```
[STATE_BEGIN Sun ONE Directory Distribution a7cc64b2f71a0452899e1c3b853ecead72027b3b]
```

Installing Using the Specification Files

To configure Directory Server and Administration Server interactively, follow these steps:

1. Verify the changes made to the silent installation specification file.

2. Perform Directory Server configuration in silent mode.

   `root# /usr/sbin/directoryserver configure -f` *dirserv-file*

   Here *dirserv-file* is the silent installation configuration file.

3. Perform Administration Server configuration in silent mode.

```
root# /usr/sbin/mpsadmserver configure -f admserv-file
```

Here *admserv-file* is the silent installation configuration file.

Proceed to "Completing the Installation Process," on page 33.

Preparing For Installation From a Compressed Archive

1. From the directory containing the software you unpacked as described in "Obtaining Directory Server Software," on page 19, run the `idsktune` utility. `idsktune` checks for appropriate patches and verifies the system is tuned to support high directory service performance.

   As super user, enter the following command:

   ```
   root# ./idsktune -q > idsktune.out
   ```

   Perform suggested changes to the system manually. `idsktune` itself makes no changes to the system.

2. Fix at least all `ERROR` conditions indicated by `idsktune`. If you do not fix `ERROR` conditions, installation may fail. Notice that `idsktune` reports as missing *all* patches recommended at the time of release and not installed on the system, even patches for packages not installed on the system.

   You may download patches from `http://sunsolve.sun.com/`.

   Refer to Chapter 5, "Tuning the Operating System" for more information.

3. (Optional) Create a user and group account for Directory Server.

   Directory Server runs as the user and group you specify during installation. Set permissions that prevent unauthorized access to the directory and to other resources on the system. Refer to "(UNIX Platforms) Users and Groups," on page 99 for more information.

4. (Optional) When installing interactively as another user, allow access to the display using the `xhost`(1) command.

   When you set the `DISPLAY` environment variable appropriately and perform installation as a user having access to the display, the installation program displays the graphical user interface by default.

   If the installation program cannot display the graphical user interface, it starts installation in command line mode.

5. Before installing using a locale other than US English, set the `LANG` environment variable to `C`.

Performing Interactive Installation From a Compressed Archive

1. Start the installation program in the directory containing the unpacked software.

   For the graphical user interface:

   ```
   root# ./setup
   ```

   For command-line interface:

   ```
   root# ./setup -nodisplay
   ```

   The first installation screen appears.

2. Follow the instructions on each screen using the work sheet you made when "Preparing Installation Information," on page 21.

NOTE To install a 32-bit Directory Server, ensure you clear the check box next to Sun ONE Directory Suite > Sun ONE Directory Server (64-bit support) in the wizard screen titled `Select Components`.

Do not install this version in the same directory as an earlier version of the Directory Server. If you must reuse the same directory location, first uninstall the earlier version. Refer to Chapter 2, "Upgrading From Previous Versions," for further information.

Performing Silent Installation From a Compressed Archive

Complete the steps in the following procedures.

Creating Specification Files

To perform a silent installation, you must first create a file containing installation specifications. For a silent installation specification file template, refer to `setup_data/typical.ins` under the directory where you unpacked the software.

NOTE Specification files may contain passwords in clear text. Protect such files with appropriate file permissions.

You may create a silent installation specification file either by editing a copy of the template file by hand, or by performing interactive configuration using the installation program.

1. Become super user.

2. Start the installation program with the `-saveState` option.

   ```
   root# ./setup -saveState filename
   ```

   to create the specification file, *filename*.

3. Perform interactive installation.

4. Adjust the specification file, *filename*, before using it to install on other systems.

   Some silent installation specification file directives, such as `FullMachineName`, depend directly on the underlying host system and so cannot be generated generically.

Silent installation specification files contain a checksum string corresponding to the build version of the install program. To reuse a silent installation specification file with a different build or release of the install program, update the checksum string in lines beginning with `[STATE_BEGIN` and `[STATE_DONE`. The updated checksum can be found in `typical.ins`. Code Example 1-1 on page 30 shows a sample checksum.

Installing Using the Specification Files

1. Verify the changes made to the installation specification file.

2. Start the installation program in silent mode.

   ```
   root# ./setup -noconsole -nodisplay -state filename
   ```

   Here *filename* is the silent installation specification file.

Completing the Installation Process

1. Ensure that access permissions for the files under *ServerRoot/*`alias` have been set to prevent access by all users other than servers you install under *ServerRoot*.

2. (Optional) If you installed from a compressed archive, add support to start Directory Server on system reboot. This support is included in the Solaris package version.

   Refer to the Solaris system administration documentation for details.

3. (Optional) Enable core file generation.

   If you have installed Directory Server as super user, but have set the user and group ID to that of another account, the Directory Server may not be able to generate a `core` file during a crash. It is strongly recommended that you plan enough space for `core` files, and allow Directory Server to generate them during a crash.

   You may administer `core` file generation using `coreadm`(1M), allowing Directory Server to generate `core` files as follows, for example:

   ```
   root# coreadm -e proc-setid
   ```

   Refer to "(UNIX Platforms) Core Files," on page 88 for further information.

4. (Optional) Many command-line scripts written in Perl can now read the bind password interactively (`-w -` option). To enable this functionality:

   a. Install the `Term::ReadKey` Perl module, available separately from CPAN.

   b. Edit each Perl script to read the bind password interactively by uncommenting the appropriate lines.

   All other Perl script functionality remains available without the `Term::ReadKey` module.

Directory Server is now minimally configured and started.

Installing on Other UNIX Systems

Proceed according to instructions in the appropriate sections:

- Preparing For Installation
- Performing Interactive Installation
- Performing Silent Installation
- Completing the Installation Process

Preparing For Installation

Proceed according to instructions in the appropriate sections:

- Instructions For All UNIX Platforms
- Additional Instructions For AIX Systems
- Additional Instructions For HP-UX Systems

Instructions For All UNIX Platforms

1. Run the `idsktune` utility, which you find in the directory containing the unpacked software. `idsktune` checks for appropriate patches and verifies the system is tuned to support high directory service performance.

   As super user, enter the following command:

   ```
   root# ./idsktune -q > idsktune.out
   ```

   Perform suggested changes to the system manually. `idsktune` itself makes no changes to the system.

2. Fix at least all `ERROR` conditions indicated by `idsktune`. If you do not fix `ERROR` conditions, installation may fail.

   Table 1-9 suggests where to look for official patches not yet installed on your system.

Table 1-9 Where to Obtain Patches, By Platform

Platform	**Browse...**
Hewlett Packard HP-UX	http://www.hp.com/support/
IBM AIX	http://www.ibm.com/support/
Red Hat Linux	http://www.redhat.com/

   Refer to Chapter 5, "Tuning the Operating System" starting on page 97 for more information.

3. (Optional) Create a user and group account for Directory Server.

   Directory Server runs as the user and group you specify during installation. Set permissions that prevent unauthorized access to the directory and to other resources on the system. Refer to "(UNIX Platforms) Users and Groups," on page 99 for more information.

4. (Optional) When installing interactively as another user, allow access to the display using the `xhost`(1) command.

   When you set the `DISPLAY` environment variable appropriately and perform installation as a user having access to the display, the installation program displays the graphical user interface by default.

   If the installation program cannot display the graphical user interface, it starts installation in command-line mode.

5. Before installing using a locale other than US English, set the `LANG` environment variable to `C`.

Additional Instructions For AIX Systems

- If you plan to use the Console, install the `X11.adt` package.

  This package is not part of the standard bundle, but may be obtained from IBM.

Additional Instructions For HP-UX Systems

1. Ensure that support for IPv6 is installed, even if you do not intend to use IPv6 interfaces with Directory Server.
2. Before installing remotely using a locale with fonts not supported for US English, ensure you can access font aliases for remote sessions.

   Refer to the operating system documentation for instructions.

Performing Interactive Installation

1. Start the installation program in the directory containing the unpacked software.

   For the graphical user interface:

   ```
   root# ./setup
   ```

   For the command-line interface:

   ```
   root# ./setup -nodisplay
   ```

   The first installation screen appears.

2. Follow the instructions on each screen using the work sheet you made when "Preparing Installation Information," on page 21.

NOTE To install a 32-bit Directory Server on platforms with 64-bit server support, ensure you clear the check box next to Sun ONE Directory Suite > Sun ONE Directory Server (64-bit support) in the wizard screen titled `Select Components`.

Do not install this version in the same directory as an earlier version of the Directory Server. If you must reuse the same directory location, first uninstall the earlier version. Refer to Chapter 2, "Upgrading From Previous Versions," for further information.

Proceed to “Completing the Installation Process,” on page 38.

Performing Silent Installation

Complete the steps in the following procedures.

Creating Specification Files

To perform a silent installation, you must first create a file containing installation specifications. For a silent installation specification file template, refer to `setup_data/typical.ins` under the directory where you unpacked the software.

NOTE Specification files may contain passwords in clear text. Protect such files with appropriate file permissions.

You may create a silent installation specification file either by editing a copy of the template file by hand, or by performing interactive configuration using the installation program.

1. Become super user.
2. Start the installation program with the `-saveState` option.

   ```
   root# ./setup -saveState filename
   ```

   to create the specification file, *filename*.
3. Perform interactive installation.
4. Adjust the specification file, *filename*, before using it to install on other systems.

   Some silent installation specification file directives, such as `FullMachineName`, depend directly on the underlying host system and so cannot be generated generically.

Silent installation specification files contain a checksum string corresponding to the build version of the install program. To reuse a silent installation specification file with a different build or release of the install program, update the checksum string in lines beginning with `[STATE_BEGIN` and `[STATE_DONE`. The updated checksum can be found in `typical.ins`. Code Example 1-1 on page 30 shows a sample checksum.

Installing Using the Specification Files

1. Verify the changes made to the installation specification file.
2. Start the installation program in silent mode.

```
root# ./setup -noconsole -nodisplay -state filename
```

Here *filename* is the silent installation specification file.

Completing the Installation Process

1. Ensure that access permissions for files under *ServerRoot*/`alias` have been set to prevent access by all users other than servers you install under *ServerRoot*.
2. (Optional) Add support to start Directory Server on system reboot.

   Refer to the operating system documentation for details.
3. (Optional) Enable core file generation.

   If you have installed Directory Server as super user, but have set the user and group ID to that of another account, the Directory Server may not be able to generate a `core` file during a crash. It is strongly recommended that you plan enough space for `core` files, and allow Directory Server to generate them during a crash.

   Refer to "(UNIX Platforms) Core Files," on page 88 for further information.
4. (Optional) Many command-line scripts written in Perl can now read the bind password interactively (`-w -` option). To enable this functionality:

   a. Install the `Term::ReadKey` Perl module, available separately from CPAN.

   a. Edit each Perl script to read the bind password interactively by uncommenting the appropriate lines.

   All other Perl script functionality remains available without the `Term::ReadKey` module.

Directory Server is now minimally configured and started.

Installing on Windows Systems

Proceed according to instructions in the appropriate sections:

- Preparing For Installation
- Performing Interactive Installation
- Performing Silent Installation
- Completing the Installation Process

Preparing For Installation

1. When installing Windows 2000, specify that the computer is a stand-alone server, not a member of any existing domain or workgroup, to reduce dependencies on network security services.
2. Apply Service Pack 3.
3. Ensure the display driver supports at least 256 colors.
4. Log on as a user with `Administrator` privileges.
5. Set the `TEMP` environment variable to a valid folder for temporary files.

Performing Interactive Installation

1. Double click `setup.exe` in the folder containing the unpacked software.

   The first installation screen appears.

2. Follow the instructions on each screen using the work sheet you made when "Preparing Installation Information," on page 21.

   Do not install this version in the same folder as an earlier version of the Directory Server. If you must reuse the same folder, first uninstall the earlier version. Refer to Chapter 2, "Upgrading From Previous Versions," for further information.

Proceed to "Completing the Installation Process," on page 40.

Performing Silent Installation

Perform the steps in the following procedures.

Creating Specification Files

To perform a silent installation, you must first create a file containing installation specifications. For a silent installation specification file template, refer to `setup_data\typical.ins` in the folder where you unpacked the software.

NOTE Specification files may contain passwords in clear text. Protect such files with appropriate file permissions.

You may create a silent installation specification file either by editing a copy of the template file by hand, or by performing interactive configuration using the installation program.

1. Log on as a user with Administrator privileges.
2. Start the installation program with the `-saveState` option.

   From the folder where you unpacked the product, enter

   Prompt>`setup -saveState` *filename*

   to create the specification file, *filename*.
3. Perform interactive installation.
4. Adjust the specification file, *filename*, before using it to install on other systems.

   Some silent installation specification file directives, such as `FullMachineName`, depend directly on the underlying host system and so cannot be generated generically.

Silent installation specification files contain a checksum string corresponding to the build version of the install program. To reuse a silent installation specification file with a different build or release of the install program, update the checksum string in lines beginning with `[STATE_BEGIN` and `[STATE_DONE`. The updated checksum can be found in `typical.ins`. Code Example 1-1 on page 30 shows a sample checksum.

Installing Using the Specification Files

1. Verify the changes made to the installation specification file.
2. Start the installation program in silent mode.

   From the folder where you unpacked the product, enter

   Prompt>`setup -noconsole -nodisplay -state` *filename*

   Here *filename* is the silent installation specification file.

Completing the Installation Process

1. Ensure that access permissions for files under *ServerRoot*`\alias` have been set to prevent access by all users other than servers you install under *ServerRoot*.
2. After installation, manually set special access permissions for the following files such that only the user and group running the Administration Server has read-write access, and all other users have no access.
   - *ServerRoot*`\admin-serv\config\adm.conf`
   - *ServerRoot*`\admin-serv\config\admpw`
   - *ServerRoot*`\admin-serv\config\magnus.conf`

- *ServerRoot*`\admin-serv\config\obj.conf`
- *ServerRoot*`\admin-serv\config\secmod.db`
- *ServerRoot*`\admin-serv\config\server.xml`

Refer to Windows help for instructions on setting special access permissions for files. This modification prevents unauthorized users from modifying Administration Server configuration data.

3. (Optional) Many command-line scripts written in Perl can now read the bind password interactively (`-w -` option). To enable this functionality:

a. Install the `Term::ReadKey` Perl module, available separately from CPAN.

b. Edit each Perl script to read the bind password interactively by uncommenting the appropriate lines.

All other Perl script functionality remains available without the `Term::ReadKey` module.

Directory Server is now minimally configured and started.

Uninstallation

Uninstallation removes the software and associated data from a computer. Directory Server becomes unavailable and you lose all settings and data.

Uninstallation removes not only server software, but also registry data stored on the system. If you delete files manually before using the uninstallation program, you may corrupt your registry. To avoid corrupting the registry, use the uninstallation program before deleting any product files manually.

NOTE You do not receive a warning before proceeding with uninstallation of your configuration directory containing configuration information under the `o=NetscapeRoot` suffix.

If you uninstall a centralized configuration directory that other directories rely on for configuration information, you cannot subsequently administer those directories.

Proceed according to the appropriate section:

- Uninstalling on Solaris Systems

- Uninstalling on Other UNIX Systems
- Uninstalling on Windows Systems

Uninstalling on Solaris Systems

How you remove Directory Server software depends on which packaging was used during the installation process, and on whether you want to interact with the uninstall program. Proceed according to instructions in the appropriate section:

- Performing Interactive Uninstallation After Installing Using Solaris Packages
- Performing Interactive Uninstallation After Installing From a Compressed Archive
- Performing Silent Uninstallation After Installing Using Solaris Packages
- Performing Silent Uninstallation After Installing From a Compressed Archive

Performing Interactive Uninstallation After Installing Using Solaris Packages

Proceed according to instructions in the appropriate sections:

- Uninstalling Previous Directory Server Versions
- Unconfiguring Administration Server
- Unconfiguring Directory Server
- Removing Packages

Uninstalling Previous Directory Server Versions

- *Important* If you are completing the upgrade of Directory Server 5.1 on a Solaris system to 5.2, and the 5.1 version was installed from `IPLT*` Solaris packages, then perform uninstallation for the 5.1 version:

  ```
  root# /usr/sbin/directoryserver.51bak uninstall
  ```

Unconfiguring Administration Server

- Delete the Administration Server configuration.

  ```
  root# /usr/sbin/mpsadmserver unconfigure
  ```

  The first uninstallation screen appears. Follow the instructions on each screen.

Unconfiguring Directory Server

- Delete the Directory Server configuration.

  ```
  root# /usr/sbin/directoryserver unconfigure
  ```

  The first uninstallation screen appears. Follow the instructions on each screen.

Removing Packages

- Using the `pkgrm`(1M) utility, remove the packages installed in "Performing Interactive Installation Using Solaris Packages," on page 24.

Performing Interactive Uninstallation After Installing From a Compressed Archive

1. In the *ServerRoot* directory, start the uninstall program.

   ```
   root# ./uninstall_dirserver
   ```

   The first uninstallation screen appears.

2. Follow the instructions on each screen.

The selected software is now removed. If the uninstallation program cannot remove all files under the *ServerRoot* directory, it displays a message. You may manually remove files remaining under *ServerRoot*.

Performing Silent Uninstallation After Installing Using Solaris Packages

1. Edit uninstall specification file, *ServerRoot*/`setup/uninstall.ins`, to include the appropriate administrator identifiers and passwords.

Code Example 1-2 Sample Uninstall Specification File

```
[STATE_BEGIN Sun ONE Directory Distribution checksum]

ConfigDirectoryAdminID = admin-user
ConfigDirectoryAdminPwd = admin-password

[STATE_DONE Sun ONE Directory Distribution checksum]
```

2. If you are completing the upgrade of Directory Server 5.1 on a Solaris system to 5.2, and the 5.1 version was installed from `IPLT*` Solaris packages, then perform uninstallation for the 5.1 version:

```
root# /usr/sbin/directoryserver.51bak uninstall -f 51-uninstaller-file
```

3. Delete the Administration Server configuration using the `unconfigure` subcommand.

```
root# /usr/sbin/mpsadmserver unconfigure -f ServerRoot/setup/uninstall.ins
```

4. Delete the Directory Server configuration using the `unconfigure` subcommand.

```
root# /usr/sbin/directoryserver unconfigure -f ServerRoot/setup/uninstall.ins
```

5. Using the `pkgrm`(1M) utility, remove the packages installed in "Performing Silent Installation Using Solaris Packages," on page 29.

You may remove remaining files manually after uninstallation completes.

Performing Silent Uninstallation After Installing From a Compressed Archive

1. Edit uninstall specification file, *ServerRoot*/`setup/uninstall.ins`, as shown in Code Example 1-2 on page 43 to include the appropriate administrator identifiers and passwords.

2. Run the uninstallation program in silent mode.

```
root# cd ServerRoot
root# ./uninstall_dirserver -noconsole -nodisplay -state setup/uninstall.ins
```

You may remove remaining files manually after uninstallation completes.

Uninstalling on Other UNIX Systems

Proceed according to instructions in the appropriate section.

Performing Interactive Uninstallation

1. In the *ServerRoot* directory, start the uninstall program.

   ```
   root# ./uninstall_dirserver
   ```

   The first uninstallation screen appears.

2. Follow the instructions on each screen.

The selected software is now removed. If the uninstallation program cannot remove all files under the *ServerRoot* directory, it displays a message. You may manually remove files remaining under *ServerRoot*.

Performing Silent Uninstallation

1. Edit uninstall specification file, *ServerRoot*`/setup/uninstall.ins`, as shown in Code Example 1-2 on page 43 to include the appropriate administrator identifiers and passwords.

2. Run the uninstallation program in silent mode.

```
root# cd ServerRoot
root# ./uninstall_dirserver -noconsole -nodisplay -state setup/uninstall.ins
```

You may remove remaining files manually after uninstallation completes.

Uninstalling on Windows Systems

Proceed according to instructions in the appropriate section.

Performing Interactive Uninstallation

1. Click `Start`, and then choose `Settings > Control Panel`.

2. Double-click `Add/Remove Programs`.

3. In the `Add/Remove Programs` window, select `Directory Server`, then click `Remove`.

4. Follow the instructions in the `Sun ONE Uninstall` window.

   If you have upgraded Directory Server, use custom uninstallation mode, and choose not to remove Basic System Libraries, which include `.dll` files shared with the new Directory Server instance.

Performing Silent Uninstallation

1. Edit uninstall specification file, *ServerRoot*`\setup\uninstall.ins`, as shown in Code Example 1-2 on page 43 to include the appropriate administrator identifiers and passwords.

2. Run the uninstallation program in silent mode.

```
Prompt>cd ServerRoot
Prompt>uninstall_dirserver -noconsole -nodisplay -state setup\uninstall.ins
```

You may remove remaining files manually after uninstallation completes.

It is strongly recommended that you reboot the Windows system after uninstallation.

Troubleshooting

Table 1-10 Common Installation Problems With Solutions

Problem	Possible Solutions
I get a message about missing libraries.	Run `idsktune` and fix at least all `ERROR` conditions, installing all recommended patches.
Installation did not work, and now I cannot uninstall. What do I do?	Removing the product registry file *unless doing so would negatively impact other products*: • `/var/sadm/install/productregistry` on Solaris systems when installing as super user • `/var/tmp/productregistry` on other UNIX systems • `productregistry` in the `system32` folder under the Windows system folder, for example `C:\WINNT\system32\productregistry`, on Windows Next, remove the partially installed files by hand before reinstalling.
Installation failed and I do not know why. Is there an installation log somewhere?	Yes. The log can be found under the following location: • On Solaris systems, `/var/sadm/install/logs` (installation as super user) or `/var/tmp` (installation as a regular user) • On other UNIX systems, `/var/tmp` • On Windows systems, `%TEMP%` folder
Clients cannot locate the server.	Try using the host name such as `dirserv`. If that does not work, make sure the server is listed in the name service you are using such as DNS, and try the fully qualified domain name such as `dirserv.example.com`. If that does not work, try using the IP address for the host such as `192.168.0.30`.
The port is in use.	If upgrading, you probably did not shut down Directory Server before you upgraded it. Shut down the old server, then manually start the upgraded one. Otherwise, another server might be using the port. Examine which ports are in use with an appropriate tool such as the `netstat`(1M) utility with the `-a` option on UNIX systems to determine which ports remain available.

Table 1-10 Common Installation Problems With Solutions *(Continued)*

Problem	Possible Solutions
An LDAP authentication error causes installation to fail.	You may have provided the incorrect fully qualified domain name during installation, such as `dirserv.nisDomain.Example.COM` instead of `dirserv.example.com`.
I have forgotten the Directory Manager DN and password.	The Directory Manager DN is recorded as the value of `nsslapd-rootdn` in *ServerRoot*`/slapd-`*serverID*`/config/dse.ldif`. The Directory Manager password is recorded as the value of `nsslapd-rootpw` in `dse.ldif`. If the password is not encrypted — we strongly recommend you encrypt it! — then it appears in `dse.ldif` in clear text, not prefixed with an encryption scheme identifier such as `{SSHA}`. If the password is encrypted, you must fix the problem manually. 1. Stop Directory Server. 2. Change the value of `nsslapd-rootpw` in `dse.ldif`, taking care not to add trailing spaces. 3. Save and close `dse.ldif`. 4. Restart the server. 5. Login as Directory Manager using the value you assigned to `nsslapd-rootpw`. 6. Set an encryption scheme for the Directory Manager password as described in the *Sun ONE Directory Server Administration Guide*, and then change the password again.

Table 1-10 Common Installation Problems With Solutions *(Continued)*

Problem	Possible Solutions
I installed the 32-bit version of the Directory Server by mistake. How do I run the 64-bit version instead?	1. Export all suffixes to LDIF as described in the *Sun ONE Directory Server Administration Guide*. 2. Remove all database files. Database files are found under the path indicated by the value of `nsslapd-directory` on `cn=config,cn=ldbm database,cn=plugins,cn=config` for the instance. 3. Install 64-bit components if you have not done so already. 4. Make *ServerRoot*`/bin/slapd/server/64/ns-slapd` executable. 5. If the operating system is running in 32-bit mode, reboot it in 64-bit mode. 6. If necessary, change cache size settings to work in 32-bit mode. Refer to Chapter 6, “Tuning Cache Sizes,” for further information. 7. Initialize all suffixes with the LDIF you exported as described in the *Sun ONE Directory Server Administration Guide*. 8. Restart the server.
I installed the 64-bit version of the Directory Server by mistake. How do I run the 32-bit version instead?	1. Export all suffixes to LDIF as described in the *Sun ONE Directory Server Administration Guide*. 2. Remove all database files. Database files are found under the path indicated by the value of `nsslapd-directory` on `cn=config,cn=ldbm database,cn=plugins,cn=config` for the instance. 3. Change the mode of *ServerRoot*`/bin/slapd/server/64/ns-slapd` so it is not executable. 4. Initialize all suffixes with the LDIF you exported as described in the *Sun ONE Directory Server Administration Guide*. 5. Restart the server.

Table 1-10 Common Installation Problems With Solutions *(Continued)*

Problem	Possible Solutions

I wrote a script to handle installation. When I tried installing using my script, the installer returned `73`, rather than `0`.
 	
What is going on here?
	The installation program return codes are as follows:
	```
	 0 - SUCCESS
	 1 - WARNING_REBOOT_REQUIRED
	 2 - WARNING_PLATFORM_SUPPORT_LIMITED
	 3 - WARNING_RESOURCE_NOT_FOUND
	 4 - WARNING_CANNOT_WRITE_LOG
	 5 - WARNING_LOCALE_NOT_SUPPORTED
	50 - ERROR_FATAL
	51 - ERROR_ACCESS
	52 - ERROR_PLATFORM_NOT_SUPPORTED
	53 - ERROR_NO_WINDOWING_SYSTEM_AVAILABLE
	54 - ERROR_RESOURCE_NOT_FOUND
	55 - ERROR_TASK_FAILURE
	56 - ERROR_USER_EXIT
	57 - ERROR_CANNOT_UPGRADE
	58 - ERROR_NOTHING_TO_DO
	59 - ERROR_IN_SERIALIZATION
	60 - ERROR_ABNORMAL_EXIT
	61 - ERROR_INCOMPATIBLE_STATEFILE
	62 - ERROR_UNKNOWN_COMMANDLINE_OPTION
	70 - ERROR_NOT_INSTALLED
	71 - PARTIALLY_UNINSTALLED
	72 - FULLY_UNINSTALLED
	73 - INSTALLED
	74 - ERROR_FAILED
	75 - ERROR_STOPPED
	76 - ERROR_STOPPED_ON_ERROR
	77 - PARTIALLY_INSTALLED
	```
	In other words, `73` indicates successful installation.

Chapter 2

Upgrading From Previous Versions

This chapter covers upgrading to Sun ONE Directory Server 5.2 from Netscape Directory Server 4.x, and from iPlanet Directory Server 5.x.

NOTE	This chapter does not explain how to upgrade from Innosoft Distributed Directory Server 4.5.1.

This chapter focuses primarily on how to migrate directory data from old servers to new servers. Refer to the *Sun ONE Directory Server Reference Manual* for details about the configuration attributes migrated from the old server to the new.

Before You Upgrade

Before upgrading, familiarize yourself with the new features available in Sun ONE Directory Server 5.2 and described in the documentation under “Suggested Reading,” on page 12. Take the opportunity to review design decisions made during implementation of existing directory services.

When Upgrading a Single Server Instance

Upgrading a server instance involves installing the new server alongside the existing server in a different *ServerRoot* using a different *serverID* and different Administration Server and Directory Server port numbers, stopping the old server, migrating configuration and directory data, and then having clients make requests to the new server.

NOTE Ensure you have sufficient disk space on the host where you run the existing server. The upgrade process requires at least enough *local disk space* to house binaries and databases for both the old and new servers, and also enough extra space to hold LDIF files containing the entries in all existing suffixes. You may estimate the local disk space required as somewhat larger than:

```
2 * (space for existing server) + (space for LDIF files)
```

The upgrade process must be performed with both servers on the same host, as data migration cannot be performed over networked drives.

Sun ONE Directory Server 5.2 provides a script to help you migrate data for a server instance. The migration script performs the following tasks in sequence:

1. Stops your existing server, and backs up the current configuration.
2. Checks schema configuration files, notifying you of differences between the standard schema configuration files and those used by your existing server.

   (4.x to 5.2 only) If an existing 4.x server uses custom schema not installed in the default location, under *ServerRoot*/`slapd-`*serverID*/`config`, you must adjust the configuration manually before migrating directory data.
3. Creates a database for each suffix stored in the old server.

   (4.x to 5.2 only) 4.x servers supported multiple suffixes per database. The migration script creates a database for each suffix on the new server.
4. Migrates server and database configuration parameters.

   4.x servers store such parameters in `slapd.conf`. 5.x servers store such parameters as entries in `dse.ldif`.

NOTE The script does not migrate data under `o=NetscapeRoot`.

When deploying a server such as Sun ONE Messaging Server that relies on data in this suffix, migrate the data in `o=NetscapeRoot` manually, or using tools provided with the server in question.

(4.x to 5.2 only) The migration script does not migrate all 4.x server parameters. In some cases, you must migrate 4.x attribute values manually. Refer to the current version of the *Sun ONE Directory Server Reference Manual* for details.

5. Migrates user-defined schema objects.
6. Migrates indexes.
7. Migrates standard server plug-ins.

   You must migrate custom plug-ins manually. At minimum, you must recompile all custom plug-ins. Refer to the *Sun ONE Directory Server Plug-In API Programming Guide* for a detailed list of plug-in API changes.

8. (5.x to 5.2 only) Migrates replication agreements.

NOTE Before replicating from a 5.2 Directory Server to a 5.1 server, set `nsslapd-schema-repl-useronly` on `cn=config` to `on`. Otherwise the 5.2 schema are pushed to the 5.1 server, preventing the 5.1 server from restarting due to duplicate objects.

9. Migrates the certificate database, and SSL parameters.
10. (5.x to 5.2 only) Migrates database links.
11. (5.x to 5.2 only) Migrates replication entries.
12. Migrates the SNMP configuration.

After the migration script completes, clients may send requests to the new server.

When Upgrading Multiple Replicated Servers

Not surprisingly, upgrading multiple servers involves upgrading each server individually. The *order* in which you upgrade servers depends, however, on the software version of existing servers and on the replication topology.

For 5.x to 5.2 upgrades, the standard process is bottom up. First you migrate the consumers. Next you upgrade the hubs. Finally you upgrade the masters. Refer to “Example 5.x Upgrade Scenario,” on page 67 for how you might do this in a particular instance.

For 4.x to 5.2 upgrades, you start by upgrading the 4.x master, then proceed with each branch of consumers being replicated from the master, starting with the consumer closest to the master in terms of replication. Refer to “Example 4.x Upgrade Scenario,” on page 60 for how you might do this in a particular instance.

If the existing environment involves multiple, replicated servers, read all relevant sections of this chapter carefully before proceeding with the upgrade. You must plan your approach fully to avoid unnecessary downtime.

For Help With Upgrades

Sun Professional Services can help you upgrade critical directory services.

Refer to `http://www.sun.com/service/sunps/sunone/` for contact information.

Upgrading a Single Server

This section describes the upgrade process from a single existing server to a single 5.2 server.

NOTE If the existing 4.x server uses custom schema, ensure the migration script can find the custom schema before migrating any data. Read "(4.x to 5.2) Handling Custom Schema," on page 55 for details.

If the migration script does not recognize the custom schema, it does not migrate the schema, but instead applies standard schema files after migrating the data to the new server. Applying the standard schema to entries that conform to custom schema may render modifications impossible, thereby making the upgraded directory read-only.

Installing the New Server

Proceed according to the instructions in Chapter 1, "Installing Sun ONE Directory Server," to install the new server on the same host as the existing server.

NOTE Ensure you have a current backup of the existing server before installing the new server.

The new server must reside in a different *ServerRoot* location than the existing server. It must also be identified by a different *serverID*.

Although you may choose to reuse most of the configuration information supplied for the original installation, do not reuse the existing port number. Instead, you may change the port number of the new server after migrating existing data.

(4.x to 5.2) Handling Custom Schema

The script provided for migrating data recognizes only those custom schema either placed in the standard `slapd.user_oc.conf` and `slapd.user_at.conf` files, or placed in other files and included in `slapd.conf` using `useroc` and `userat` directives. If, for example, custom schema are included directly in `slapd.at.conf` or `slapd.oc.conf`, the migration script does not recognize them.

Perform the following steps before proceeding with the upgrade.

1. Compare `slapd.at.conf` or `slapd.oc.conf` with the standard files provided under *ServerRoot*`/bin/slapd/install/version4/` of the new server, transcribing custom schema elements to `slapd.user_oc.conf` and `slapd.user_at.conf` files.

   If the custom object classes have inheritance relationships, ensure that superior object classes precede others in the schema configuration file.

2. If custom attributes were added to standard object classes in `slapd.oc.conf`, create a new object class including the attributes in `slapd.user_oc.conf`, and add the new object class to every entry in the existing directory that uses the custom attributes.

3. Include the `slapd.user_oc.conf` and `slapd.user_at.conf` files in the `slapd.conf` file for the existing server using the `useroc` and `userat` directives, placing the new directives adjacent to include statements for other files.

At this point, all custom schema used by the existing server should reside in `slapd.user_oc.conf` or `slapd.user_at.conf`, and `slapd.conf` should include the files using the `useroc` and `userat` directives.

Migrating Existing Data

After handling custom schema, perform the following steps to migrate existing data to a new server.

1. If you intend to initialize replication on the new Directory Server offline from files, obtain the files before proceeding.

   Refer to the *Sun ONE Directory Server Administration Guide* for instructions on exporting Directory Server data.

2. Ensure the new Directory Server is running.

3. Work as a user having the right to start, stop, and run database export and import on both the old and new servers.

   For example, become super user or log on as `Administrator`.

4. Set environment variables as shown in Table 2-1.

Table 2-1 Environment Variables for Migration

Variable	**Value**
`PATH`	(UNIX) *ServerRoot*`/bin/slapd/admin/bin:$PATH`
	(Windows) *ServerRoot*`/bin/slapd/admin/bin;%PATH%`
`PERL5LIB`	*ServerRoot*`/bin/slapd/admin/bin`

5. Run the migration script under the new server instance:

```
# cd ServerRoot/bin/slapd/admin/bin
# perl migrateInstance5 -p port52 -D "cn=directory manager" -w password -o oldServ -n newServ
```

   Here, *oldServ* represents the full path to the old server instance, such as `/usr/iplanet/servers/slapd-ldap` or `/usr/iplanet/ds5/slapd-ldap`, and *newServ* represents the full path to the new server instance, such as `/var/ds/v5.2/slapd-dirserv`.

The script generates output as it proceeds. You may choose to redirect this output to a file for review after migration completes.

Retire the old server only after migrating existing data to the new server.

(4.x to 5.2) Creating Replication Agreements

If an existing 4.x server is involved in replication, upgrading involves recreating replication agreements after migrating data. Read “(4.x to 5.2) Upgrading Replicated Servers,” on page 57 before proceeding with the upgrade process.

Refer to the *Sun ONE Directory Server Administration Guide* for instructions on configuring replication for 5.2 servers.

(Optional) Reusing the Existing Port Number

After migrating data from the old server to the new, you may choose to retire the old server and have the new server listen on the same port as the old. Using the same port may allow client applications to continue operating without changing their configurations.

Refer to the *Sun ONE Directory Server Administration Guide* for instructions on changing the server port. Be sure to stop the old server before the new server starts to listen on the old port.

(4.x to 5.2) Upgrading Replicated Servers

When upgrading replicated 4.x servers, start by replicating to a new master, and then proceed branch by branch through the replication topology. This approach limits the volume of server synchronization traffic.

NOTE Refer to the *Sun ONE Directory Server Administration Guide* for detailed instructions concerning replication configuration and initialization.

Preparing the New Master

During the upgrade, the 5.2 server is configured as a master but functions as a legacy consumer in the 4.x topology. After the upgrade, the 4.x consumer capability is disabled, and the new server functions as a master in the 5.2 topology.

This procedure calls for manual configuration of the new master server. You may therefore install the new master on a different host than the existing master.

1. Proceed according to the instructions in Chapter 1, “Installing Sun ONE Directory Server,” to install the new server.
2. Manually reproduce the configuration of the 4.x master on the new server.
3. Make the new server a master (for the 5.2 topology).

   Refer to the *Sun ONE Directory Server Administration Guide* for instructions.

4. Make the new server as a legacy consumer of the 4.x master (for the 4.x topology).

   Again, refer to the *Sun ONE Directory Server Administration Guide* for instructions.

5. Initialize replication from the 4.x master to the new server.

   The process is described in Chapter 13, "*Managing Replication*," of the *Netscape Directory Server Administration Guide*. Refer to the section entitled, "*Manual Consumer Initialization*."

You may now upgrade the consumers.

Upgrading the Consumers

This procedure outlines the approach. Refer to subsequent procedures for details.

1. Upgrade all branches in the 4.x topology.
2. Add additional servers to the 5.2 topology as required.
3. Disable the legacy consumer agreement on the new master to sever the new topology from the old.

Upon completion of this procedure, the update process is complete.

Upgrading a Branch

Think of the existing 4.x replication topology as a tree with the master as the root element. Here, a *branch* denotes a set of replicated servers in that tree for which the flow of replication originates at the root node supplier, continues out through consumers in the midst of the tree, and finally arrives at leaf node consumer servers.

Upgrading a branch consists of replacing all old servers in the branch with new servers, working from the top down.

NOTE While you upgrade a server, replication flow stops to all downstream servers in the branch. Consider redirecting client requests to another branch during the upgrade.

1. Proceed according to the instructions under "Upgrading a Single Server," on page 54 to upgrade the top server in the branch.

   This cuts replication flow to the branch, temporarily bringing replication updates on downstream servers in the branch to a halt.

2. Configure the replication agreement on the new server in the 5.2 branch to receive updates from a 5.2 server closer in the replication topology to the new master.

   For example, configure the top server in the new branch to receive updates from the 5.2 master.

3. Initialize replication from the 5.2 supplier to the new 5.2 server.

   Depending on network capacity and volume of directory data compared to updates, offline initialization may be faster than online initialization.

4. Apply Step 1, Step 2, then Step 3 recursively along the branch until you have completed the steps for all leaf consumers.

Refer to the *Sun ONE Directory Server Administration Guide* for instructions on configuring replication agreements and initializing replication.

At this point, the update process is complete for the branch. Repeat the procedure for the remaining 4.x branches.

Adding Additional Servers

After completing the upgrade from the 4.x topology to the 5.2 topology, you may add additional masters, hubs, and consumers as required for the new topology.

Perform the following steps for each additional server.

1. Proceed according to the instructions under Chapter 1, "Installing Sun ONE Directory Server," to install the new server.

2. Adjust replication agreements on the new server to fit the planned topology.

   Refer to the *Sun ONE Directory Server Administration Guide* for instructions.

3. Initialize replication on the new server.

   Again, refer to the *Sun ONE Directory Server Administration Guide* for instructions.

Example 4.x Upgrade Scenario

Consider an upgrade for a 4.x master replicating to two branches, one with single consumer, one with hub supplying two consumers. This section shows the steps performed to upgrade to a new multi-master topology.

Figure 2-1 shows the 4.x topology before the upgrade.

Figure 2-1 Existing 4.x Topology Example

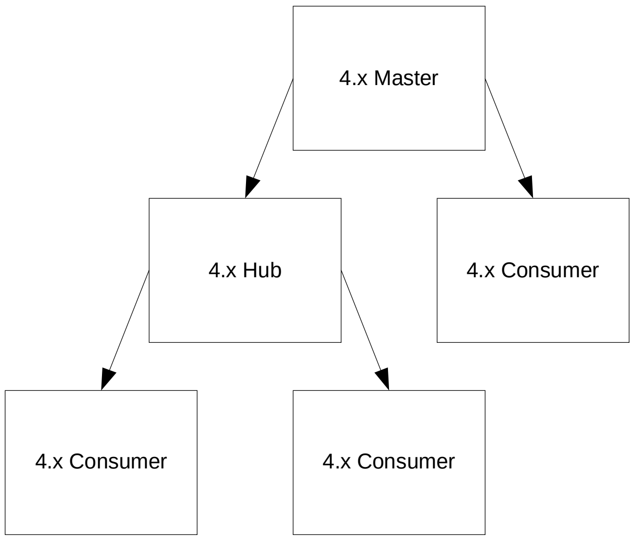


Figure 2-2 shows the addition of a 5.2 master that also functions as a legacy consumer of the 4.x master.

Figure 2-2 Example 4.x Topology with Additional New Server

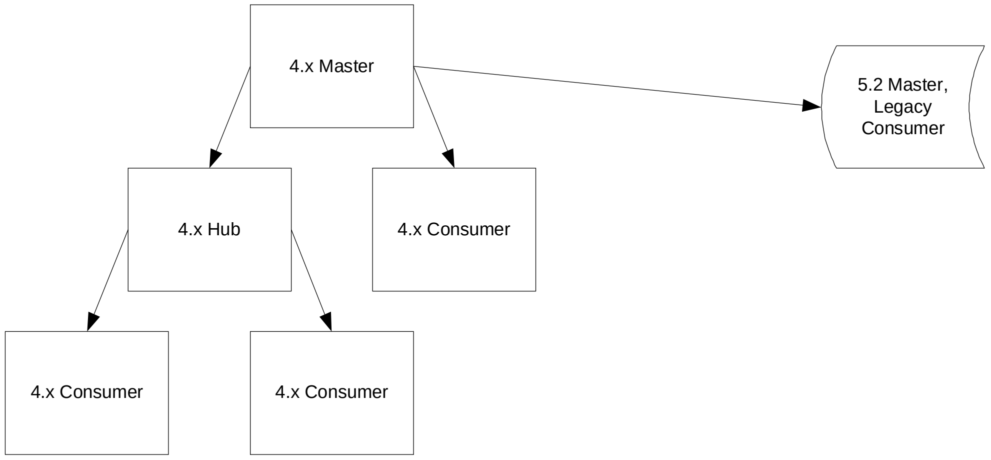


Figure 2-3 shows the first step in replacing a 4.x branch.

Notice the entire branch stops receiving replication updates during the upgrade. This interruption starts when the upstream 4.x consumer is stopped for upgrade, and ends when you restart the 4.x consumer.

As mentioned in the instructions, you may choose to direct client requests to consumers on another branch if clients require the very latest updates available.

Figure 2-3 Example 4.x Branch During Upgrade - Step 1

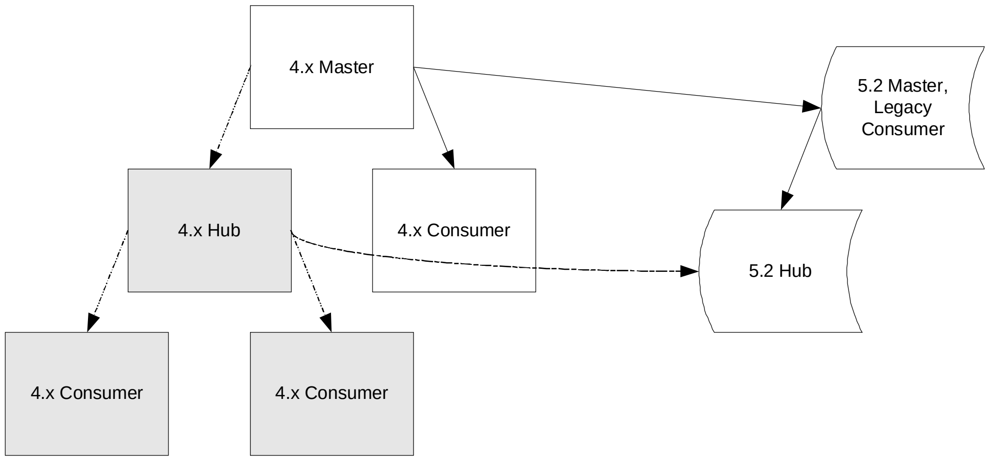


Figure 2-4 shows the next step in replacing a 4.x branch.

Figure 2-4 Example 4.x Branch During Upgrade - Step 2

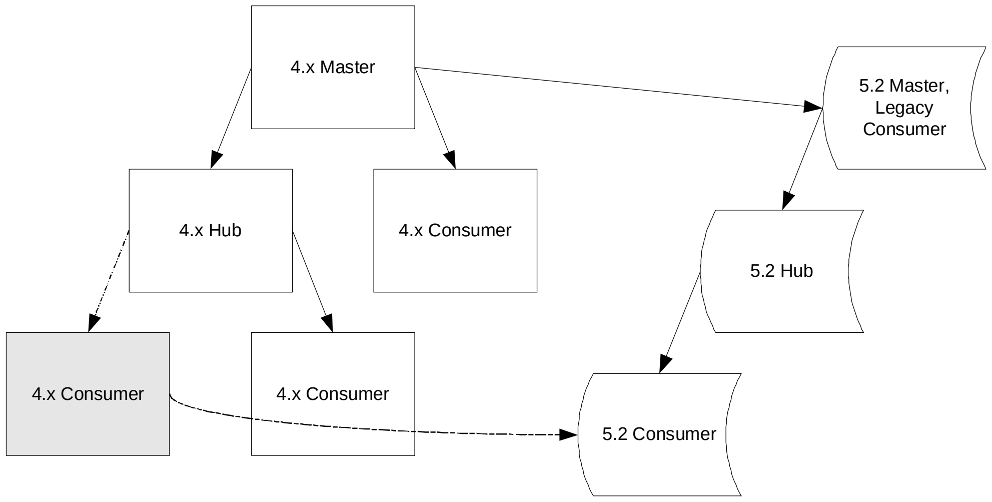


Figure 2-5 shows the next step in replacing a 4.x branch.

Figure 2-5 Example 4.x Branch During Upgrade - Step 3

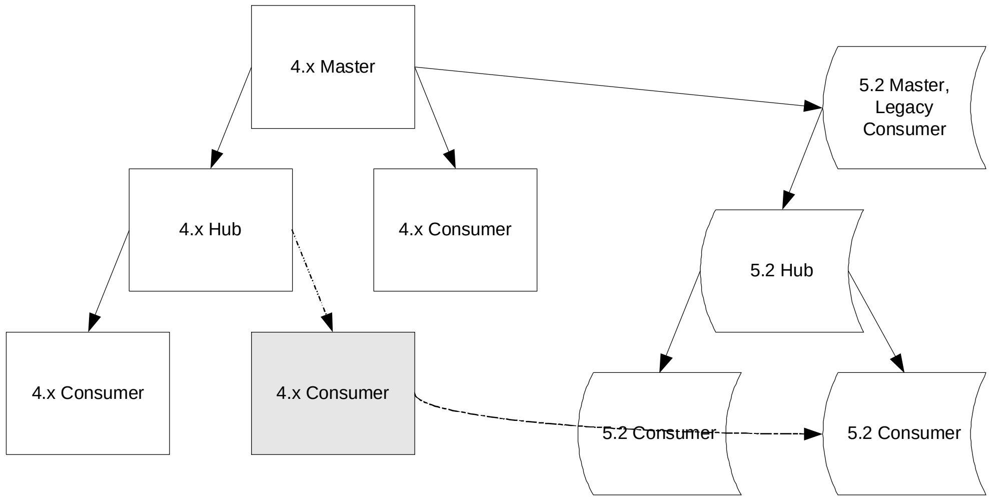


Figure 2-6 shows replacement of the other 4.x branch.

Figure 2-6 Example 4.x Branch During Upgrade - Next Branch

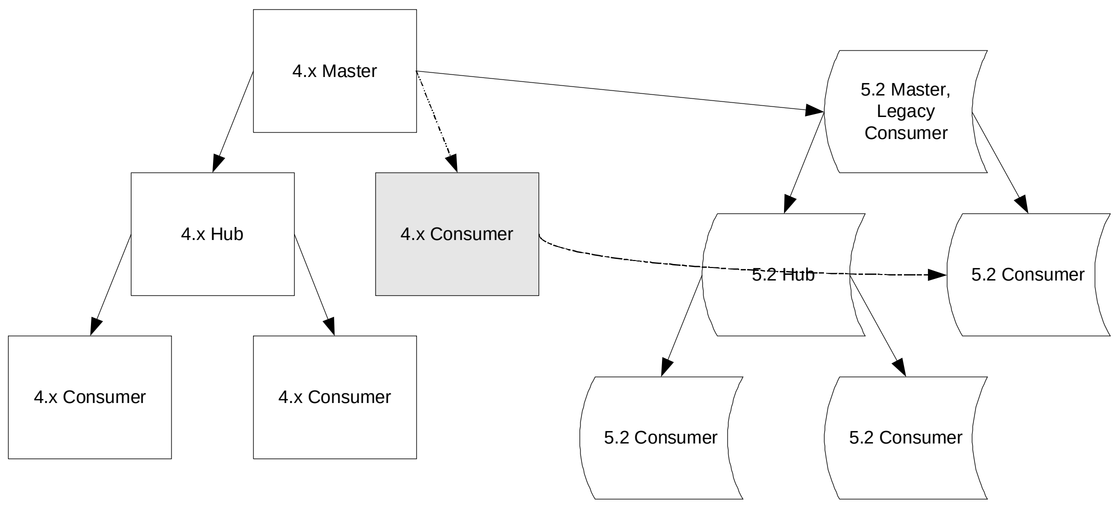


Figure 2-7 shows the two topologies side by side.

Figure 2-7 Example of 4.x and 5.2 Topologies During Upgrade

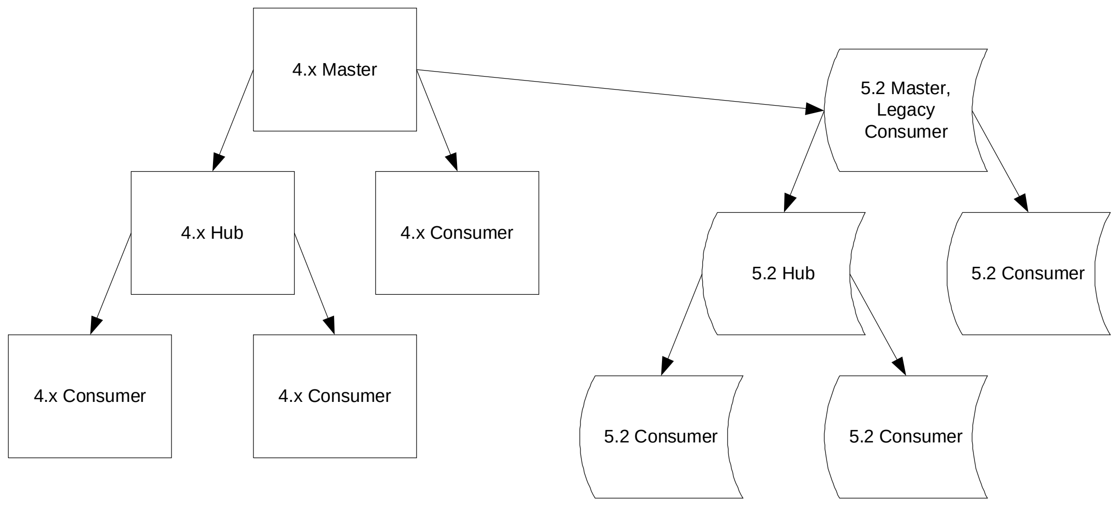
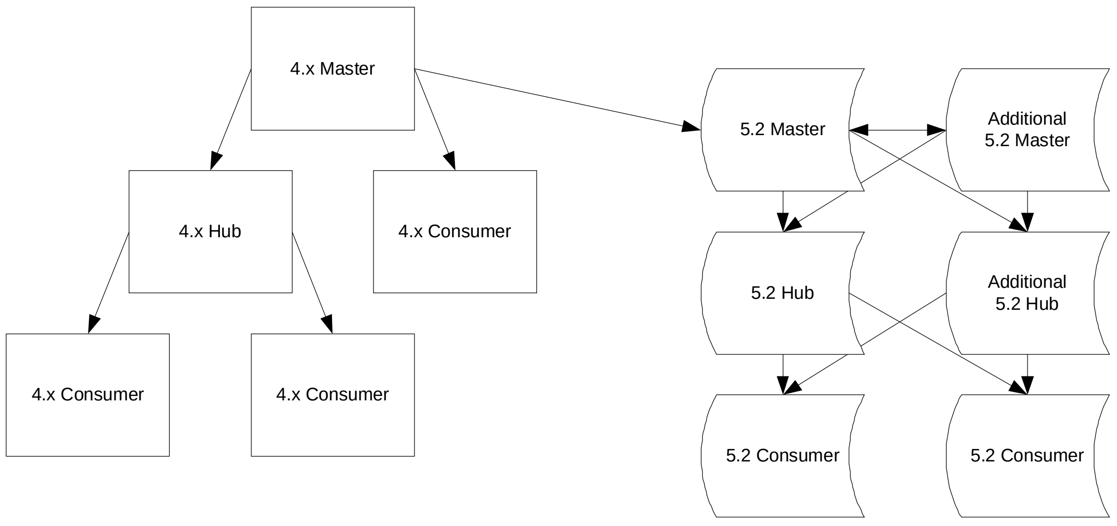


Figure 2-8 shows the addition of a master, a hub and additional replication agreements to the new topology.

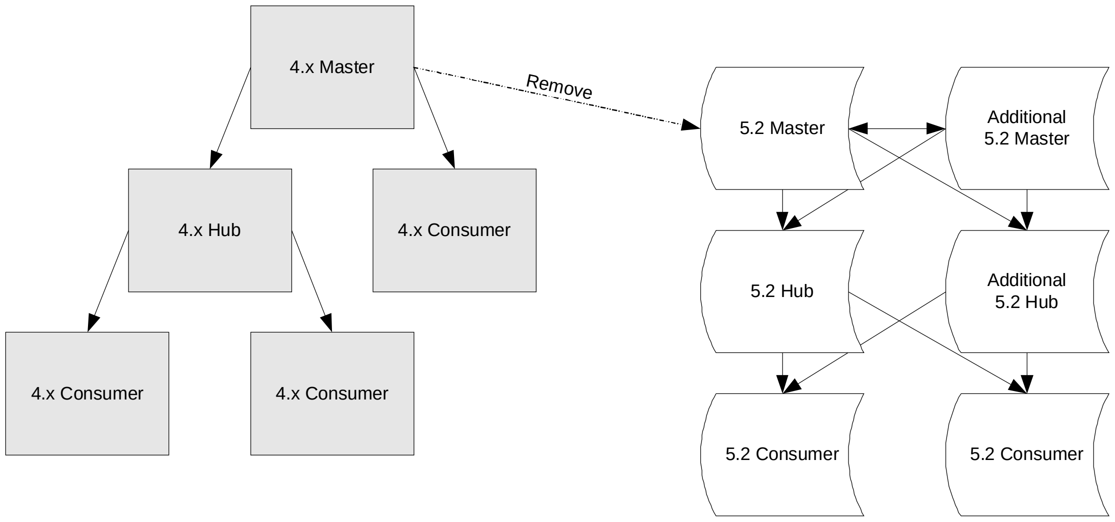
Figure 2-8 Adding Servers to the 5.2 Topology



You may also add additional servers after completing the upgrade process.

Figure 2-9 shows removal of the replication agreement from the old 4.x master to the new 5.2 master.

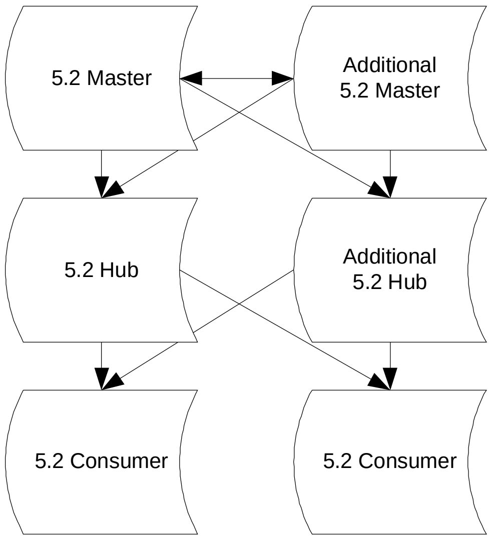
Figure 2-9 Removing the Replication Agreement



After redirecting client requests and removing the replication agreement, you may disable the 4.x servers.

Figure 2-10 shows the resulting 5.2 topology.

Figure 2-10 Resulting 5.2 Topology



Client requests are now directed to the 5.2 topology.

(5.x to 5.2) Upgrading Replicated Servers

When upgrading replicated 5.x servers, you typically start with the consumers, continue with the hubs, and finish with the masters. This bottom-up approach involves interrupting only one server at a time, rather than interrupting an entire branch of the replication topology. The approach also helps you avoid potential custom schema synchronization issues between masters and consumers.

NOTE The procedure described here applies the standard approach to upgrading a 5.x topology.

If, however, this bottom up approach fails to meet your specific requirements, then plan a different approach.

Upgrading 5.x Servers

1. For each consumer in the existing topology, proceed according to the instructions under “Upgrading a Single Server,” on page 54 to upgrade the consumer.
2. For each hub in the existing topology, proceed according to the same instructions to update the hub.
3. For each master in the existing topology, proceed according to the same instructions to update the master.

Adding Additional Servers

After completing the upgrade from the 5.x topology to the 5.2 topology, you may add additional masters, hubs, and consumers as required for the new topology.

Perform the following steps for each additional server.

1. Proceed according to the instructions in Chapter 1, “Installing Sun ONE Directory Server,” to install the new server.
2. Adjust replication agreements on the new server to fit the planned topology.
3. Initialize replication on the new server.

Refer to the *Sun ONE Directory Server Administration Guide* for instructions on configuring replication agreements and initializing replication.

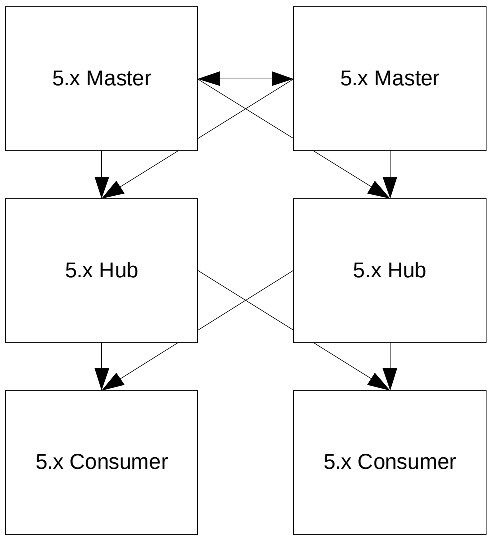
Upon completion of this procedure, the update process is complete. Clients may begin using servers in the upgraded replication topology.

Example 5.x Upgrade Scenario

Consider an upgrade for 5.x dual masters replicating to two hubs supplying two consumers. This section shows the steps performed to upgrade the topology to use 5.2 servers.

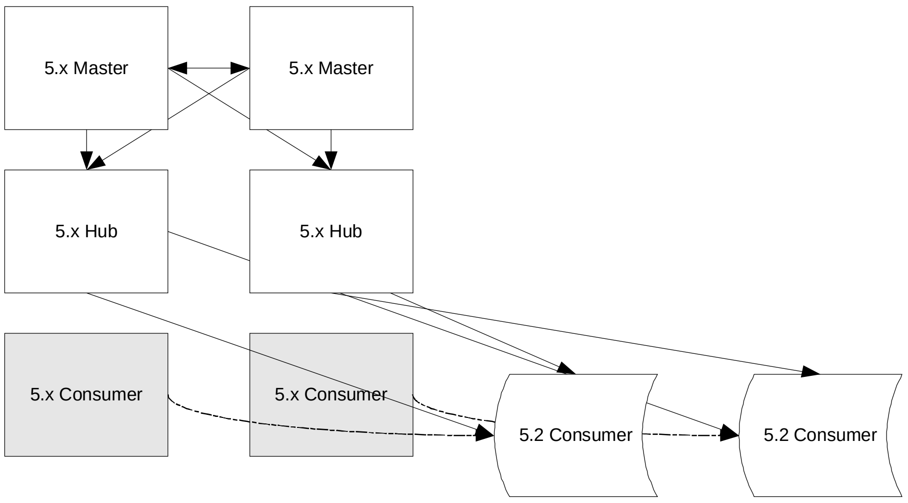
Figure 2-11 shows the 5.x topology before the upgrade.

Figure 2-11 Existing 5.x Topology Example



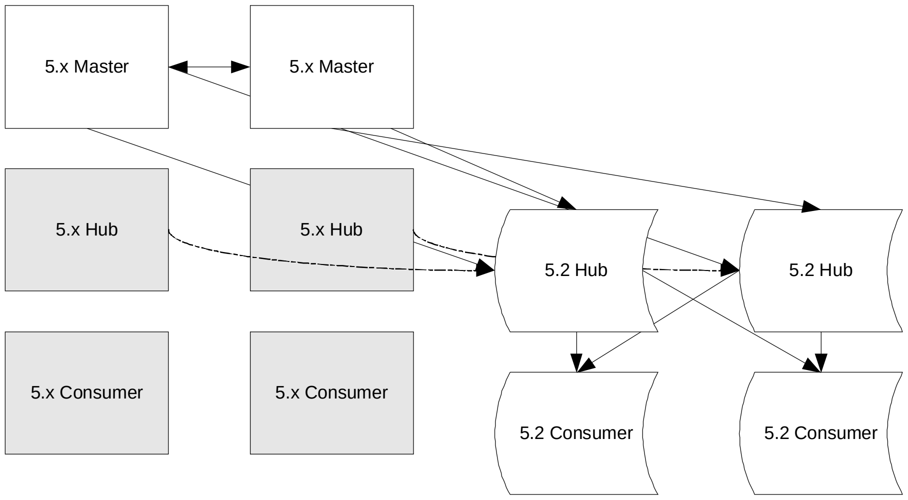
The first step involves upgrading consumers. Figure 2-12 shows the resulting topology.

Figure 2-12 Example 5.x Consumer Upgrade Step



The next step involves upgrading hubs. Figure 2-13 shows the results.

Figure 2-13 Example 5.x Hub Upgrade Step



The next step involves upgrading masters. Figure 2-14 shows the results.

Figure 2-14 Example 5.x Master Upgrade - Step 3

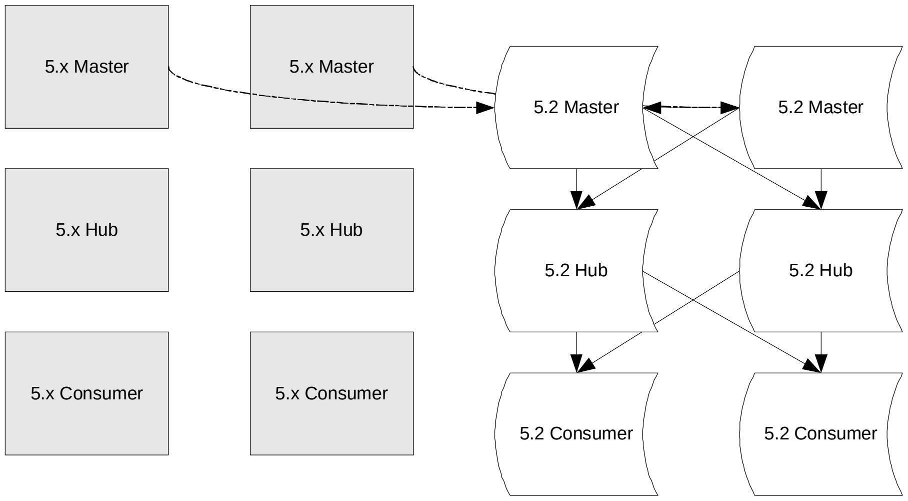
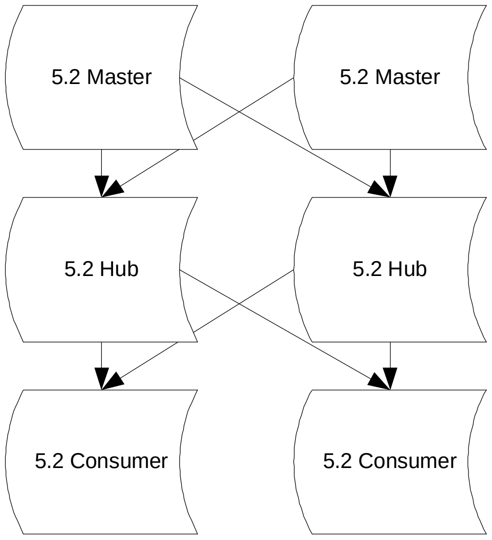


Figure 2-15 shows the 5.2 topology following the upgrade. At this point, servers in the old topology may be retired, and new servers added to the 5.2 topology.

Figure 2-15 Example 5.2 Topology after Upgrading



Client requests are now directed to the 5.2 topology.

Part 2

Tuning

Chapter 3

Top Tuning Tips

Tuning performance implies modifying the default configuration to reflect specific deployment requirements.

This guide describes how to tune a single Directory Server instance. It is assumed here that your overall directory service design including the replication topology is complete, and that you use the information here to tune the Directory Server instances to meet the design requirements. If you have not yet completed the overall directory service design, refer to the *Sun ONE Directory Server Deployment Guide* for suggestions on how to do so.

Tuning performance takes time, effort, and thought as reflected in Table 3-1.

Table 3-1 Tuning Process

Phase	Description
Define goals	Define specific, measurable objectives for tuning, based on deployment requirements. Consider questions such as: • Which applications use Directory Server? • Is the system dedicated to Directory Server? Does it run other applications? If so, which other applications? • How many *entries* does the deployment call for? How large are such entries? • How many *searches* per second must the Directory Server support? What types of searches are expected? • How many *updates* per second must the Directory Server support? What types of updates are expected? • What sort of peak update and search rates are expected? What sort of average rates are expected? • Does the deployment call for repeated bulk import initialization on this system? If so, how often are imports performed? How many entries are imported at a time? What types of entries? Must initialization be performed online with the server running? This list is not exhaustive. Ensure yours is.
Select methods	Determine how you plan to implement tuning optimizations and how you plan to measure and analyze them. Can you change the hardware configuration of the system? Are you limited to using existing hardware, tuning only the underlying operating system and Directory Server itself? How can you simulate other applications? How should you generate representative data samples for testing? How should you measure results? How should you analyze results?
Perform tests	Carry out the tests planned. For large and complex deployments, this phase may take considerable time.
Verify results	Check whether the potential optimizations tested reach the goals defined at the outset of the process. If they reach the goals, document the results. If they do not reach the goals, profile and monitor the Directory Server you are tuning.

Table 3-1 Tuning Process *(Continued)*

Phase	**Description**
Profile and monitor	Profile and monitor the behavior of Directory Server after applying the potential modifications. Collect measurements of all relative behavior.
Plot and analyze	Plot and analyze the behavior observed while profiling and monitoring. Attempt to find evidence and patterns that suggest further tests. You may need to go back to the profiling and monitoring phase to collect more data.
Tweak and tune	Apply further potential optimizations suggested by your analysis of measurements. Return to the phase of performing tests.
Document results	Once the optimizations applied reach the goals defined at the outset of the process, document them well so they can be easily reproduced.

This chapter lists basic recommendations that apply almost every time you tune a Directory Server instance. Although the recommendations presented here are in general valid, avoid the temptation to apply them without understanding how they impact the specific deployment at hand. This chapter is intended as a checklist, not a cheat sheet.

1. Adjust cache sizes.

   Ideally, the server has enough available physical memory to hold all caches used by Directory Server. In that case, set the entry cache size large enough to hold all entries in the directory, and set the database cache size large enough to hold all indexes.

   Refer to Chapter 6, "Tuning Cache Sizes," for more information.

2. Optimize indexing.

a. Remove unnecessary indexes and add additional indexes to support expected requests.

   From time to time, it may become necessary to add additional indexes that support requests from new applications. It is possible to add, remove, and modify indexes while Directory Server is running, with the limitation that existing data are only indexed progressively from that point forward.

   Refer to “Benefits: How Searches Use Indexes,” on page 126 and “Costs: How Updates Affect Indexes,” on page 127 for more information.

b. Allow only indexed searches.

   Unindexed searches can have a strong negative impact on server performance and may consume significant server resources. Consider adding indexes to support specific searches applications may perform, and forcing the server to reject unindexed searches.

   Refer to “Allowing Only Indexed Searches,” on page 134 for more information.

c. Adjust the maximum length of index lists.

   Refer to “Limiting Index List Length,” on page 135 for more information.

3. Tune the underlying operating system.

   Refer to Chapter 5, “Tuning the Operating System,” for more information.

4. Adjust operational limits.

   Adjustable operational limits prevent Directory Server from devoting inordinate resources to any single operation. Consider assigning unique bind DNs to client applications requiring increased capabilities, then setting resource limits specifically for these unique bind DNs.

   Refer to Chapter 9, “Managing Use of Other Resources,” for more information.

5. Disable unnecessary logging.

   Disk access is much slower than memory access. Writing frequently to log files on disk can have strong negative impact on performance. If possible, eliminate disk writes by turning access, error, and audit logging off when not required. At minimum, attempt to reduce the impact of logging by putting log files on separate disks using separate controllers.

   Refer to Chapter 8, “Tuning Logging,” for more information.

6. Distribute disk activity.

Especially for deployments supporting large numbers of updates, Directory Server can be extremely disk I/O intensive. If possible, consider spreading the load across multiple disks using separate controllers.

Refer to “Sizing Disk Subsystems,” on page 85 for more information.

Chapter 4

Hardware Sizing

Appropriate hardware sizing is a critical component of directory service planning and deployment. When sizing hardware, the amount of memory available and the amount of local disk space available are of key importance.

NOTE For best results, install and configure a test system with a subset of entries representing those used in production. You can then use the test system to approximate the behavior of the production server.

When optimizing for particular systems, ensure you understand how system buses, peripheral buses, I/O devices, and supported file systems work so you can take advantage of I/O subsystem features when tuning these to support Directory Server.

This chapter suggests ways of estimating disk and memory requirements for a Directory Server instance. It also touches on network and SSL accelerator hardware requirements.

Suggested Minimum Requirements

Table 4-1 proposes minimum memory and disk space requirements for installing and using the software in a production environment.

Minimum requirements for specified numbers of entries may in fact differ from those provided in Table 4-1. Sizes here reflect relatively small entries, with indexes set according to the default configuration, and with caches minimally tuned. If entries include large binary attribute values such as digital photos, or if indexing or caching is configured differently, then revise minimum disk space and memory estimates upward accordingly.

Table 4-1 Minimum Disk Space and Memory Requirements

Required for...	Free Local Disk Space	Free RAM
Unpacking product	At least 125 MB	-
Product installation	At least 200 MB	At least 256 MB
10,000-250,000 entries	Add at least 3 GB	Add at least 256 MB
250,000-1,000,000 entries	Add at least 5 GB	Add at least 512 MB
Over 1,000,000 entries	Add 8 GB or more	Add 1 GB or more

Minimum disk space requirements include 1 GB devoted to access logs. By default, Directory Server is configured to rotate through 10 access log files (`nsslapd-accesslog-maxlogsperdir` on `cn=config`) each holding up to 100 MB (`nsslapd-accesslog-maxlogsize` on `cn=config`) of messages. Volume for error and audit logs depends on how Directory Server is configured. Refer to the *Sun ONE Directory Server Administration Guide* for details on configuring logging.

Minimum Available Memory

Minimum memory estimates reflect memory used by an instance of Directory Server in a typical deployment. The estimates do not account for memory used by the system and by other applications. For a more accurate picture, you must measure memory use empirically. Refer to “Sizing Physical Memory,” on page 81 for details.

As a rule, the more available memory, the better.

Minimum Local Disk Space

Minimum local disk space estimates reflect the space needed for an instance of Directory Server in a typical deployment. Experience suggests that if directory entries are large, the space needed is at minimum four times the size of the equivalent LDIF on disk. Refer to “Sizing Disk Subsystems,” on page 85 for details.

Do *not* install the server or any data it accesses on network disks. Sun ONE Directory Server software does not support the use of network attached storage via NFS, AFS, or SMB. Instead, all configuration, log, database, and index files must reside on local storage at all times, even after installation.

On Windows systems, format drives as NTFS rather than FAT. FAT is not supported for use with Directory Server. NTFS allows access controls to be set on files and directories.

Minimum Processing Power

High volume systems typically employ multiple, high-speed processors to provide appropriate processing power for multiple simultaneous searches, extensive indexing, replication, and other features. Refer to “Sizing for Multiprocessor Systems,” on page 94 for details.

Minimum Network Capacity

Testing has demonstrated that 100 Mbit Ethernet may be sufficient for even service provider performance, depending on the maximum throughput expected. You may estimate theoretical maximum throughput as follows:

```
max. throughput = max. entries returned/second x average entry size
```

Imagine for example that a Directory Server must respond to a peak of 5000 searches per second for which it returns 1 entry each with entries having average size of 2000 bytes, then the theoretical maximum throughput would be 10 MB, or 80 Mbit. 80 Mbit is likely to be more than a single 100 Mbit Ethernet adapter can provide. Actual observed performance may vary.

If you expect to perform multi-master replication over a wide area network, ensure the connection provides sufficient throughput with minimum latency and near-zero packet loss.

Refer to “Sizing Network Capacity,” on page 94 for more information.

Sizing Physical Memory

Directory Server stores information using database technology. As is the case for any application relying on database technology, adequate fast memory is key to optimum Directory Server performance. As a rule, the more memory available, the more directory information can be cached for quick access. In the ideal case, each server has enough memory to cache the entire contents of the directory at all times. As Sun ONE Directory Server 5.2 supports 64-bit memory addressing, cache sizes are no longer limited to several gigabytes. Instead, it is now theoretically possible to handle total cache sizes of over 1.5 terabytes on 64-bit architectures.

NOTE When deploying Directory Server in a production environment, configure cache sizes well below theoretical process limits, leaving appropriate resources available for general system operation.

Estimating memory size required to run Directory Server involves estimating the memory needed both for a specific Directory Server configuration, and for the underlying system on which Directory Server runs.

Sizing Memory for Directory Server

Given estimated configuration values for a specific deployment, you can estimate physical memory needed for an instance of Directory Server. Table 4-2 summarizes the values used for the calculations in this section.

Table 4-2 Values for Sizing Memory for Directory Server

Value	Description[1]
`nsslapd-cachememsize`	Entry cache size for a suffix An entry cache contains formatted entries, ready to be sent in response to a client request. One instance may handle several entry caches.
`nsslapd-dbcachesize`	Database cache size The database cache holds elements from databases and indexes used by the server.
`nsslapd-import-cachesize`	Database cache size for bulk import Import cache is used only when importing entries. You may be able to avoid budgeting extra memory for import cache, instead reusing memory budgeted for entry or database cache if you perform *only* offline imports.
`nsslapd-maxconnections`	Maximum number of connections managed.
`nsslapd-threadnumber`	Number of operation threads created at server startup

1. For complete descriptions, refer to the *Sun ONE Directory Server Reference Manual*.

To estimate approximate memory size, perform the following steps.

1. Estimate the base size of the server process, `slapdBase`.

```
slapdBase     = 75 MB +(nsslapd-threadnumber x 0.5 MB) +(nsslapd-maxconnections x 0.5 KB)
```

2. Determine the sum of entry cache sizes, `entryCacheSum`.

entryCacheSum = $\text{Sum}_{\text{all entry caches}}$(nsslapd-cachememsize)

3. Determine the total size for all caches, `cacheSum`.

```
cacheSum      = entryCacheSum + nsslapd-dbcachesize + nsslapd-import-cachesize
```

4. Determine the total size for the Directory Server process, `slapdSize`.

```
slapdSize     = slapdBase + cacheSum
```

   You may use utilities such as `pmap`(1) on Solaris systems or the Windows Task Manager to measure physical memory used by Directory Server.

5. Estimated memory needed to handle incoming client requests, `slapdGrowth`.

```
slapdGrowth   = 20% x slapdSize
```

   As a first estimate, we assume 20 percent overhead for handling client requests. The actual percentage may depend on the characteristics of your particular deployment. Validate this percentage empirically before putting Directory Server into production.

6. Determine total memory size for Directory Server, `slapdTotal`.

```
slapdTotal    = slapdSize + slapdGrowth
```

   For large deployments involving 32-bit servers, `slapdTotal` may exceed the practical limit of about 3.4 GB, and perhaps even the theoretical process limit of about 3.7 GB. In this case, you may choose either to tune caching as suggested in Chapter 6, “Tuning Cache Sizes,” to work within the limits of the system, or to use a 64-bit version of the product.

Sizing Memory for the Operating System

Estimating the memory needed to run the underlying operating system must be done empirically, as operating system memory requirements vary widely based on the specifics of the system configuration. For this reason, consider tuning a representative system for deployment as described in Chapter 5, “Tuning the Operating System,” before attempting to estimate how much memory the underlying operating system needs. After tuning the system, monitor memory use to arrive at an initial estimate, `systemBase`. You may use utilities such as `sar`(1M) on Solaris systems or the Task Manager on Windows to measure memory use.

NOTE For top performance, dedicate the system running Directory Server to this service only.

If you must run other applications or services, monitor the memory they use as well when sizing total memory required.

Additionally, allocate memory for general system overhead and normal administrative use. A first estimate for this amount, `systemOverhead`, should be at least several hundred megabytes, or 10 percent of the total physical memory, whichever is greater. The goal is to allocate enough space for `systemOverhead` that the system avoids swapping pages in and out of memory while in production.

The total memory needed by the operating system, `systemTotal`, can then be estimated as follows.

```
systemTotal = systemBase + systemOverhead
```

Sizing Total Memory

Given `slapdTotal` and `systemTotal` estimates from the preceding sections, estimate the total memory needed, `totalRAM`.

```
totalRAM = slapdTotal + systemTotal
```

Notice `totalRAM` is an *estimate* of the total memory needed, including the assumption that the system is dedicated to the Directory Server process, and including estimated memory use for all other applications and services expected to run on the system.

Dealing With Insufficient Memory

In many cases, it is not cost effective to provide enough memory to cache all data used by Directory Server.

At minimum, equip the server with enough memory that running Directory Server does not cause constant page swapping. Constant page swapping has a strong negative performance impact. You may use utilities such as `vmstat`(1M) on Solaris and other systems to view memory statistics before and after starting Directory Server and priming the entry cache. Unsupported utilities available separately such as `MemTool` for Solaris systems can be useful in monitoring how memory is used and allocated when applications are running on a test system.

If the system cannot accommodate additional memory, yet you continue to observe constant page swapping, reduce the size of the database and entry caches. Running out of swap space can cause the Directory Server to crash.

Refer to Chapter 6, “Tuning Cache Sizes”, for a discussion of the alternatives available when providing adequate physical memory to cache all directory data is not an option.

Sizing Disk Subsystems

Disk use and I/O capabilities can strongly impact performance. Especially for a deployment supporting large numbers of modifications, the disk subsystem can become an I/O bottleneck. This section offers recommendations for estimating overall disk capacity for a Directory Server instance, and for alleviating disk I/O bottlenecks.

Refer to Chapter 8, “Tuning Logging”, for more information on alleviating disk I/O bottlenecks.

Sizing Directory Suffixes

Disk space requirements for a suffix depend not only on the size and number of entries in the directory, but also on the directory configuration and in particular how the suffix is indexed. To gauge disk space needed for a large deployment, perform the following steps:

1. Generate LDIF for three representative sets of entries like those expected for deployment, one of 10,000 entries, one of 100,000, one of 1,000,000.

   Generated entries should reflect not only the mix of entry types (users, groups, roles, entries for extended schema) expected, but also the average size of individual attribute values, especially if single large attribute values such as `userCertificate` and `jpegPhoto` are expected.

2. Configure an instance of Directory Server as expected for deployment.

   In particular, index the database as you would for the production directory. If you expect to add indexes later, expect to have to add space for those indexes as well.

3. Load each set of entries, recording the disk space used for each set.
4. Graph the results to extrapolate estimated suffix size for deployment.

5. Add extra disk space to compensate for error and variation.

Disk space for suffixes is only part of the picture; you must also consider how Directory Server uses disks.

How the Directory Server Uses Disks

Directory suffixes are part of what Directory Server stores on disk. A number of other factors affecting disk use may vary widely depending even on how Directory Server is used after deployment and so are covered here in general terms. Refer to the *Sun ONE Directory Server Administration Guide* for instructions on configuring the items discussed here.

Directory Server Binaries

You need approximately 200 MB disk space to install this version of Directory Server. This estimate is not meant to include space for data, but only for the product binaries.

Event Logging

Disk use estimates for log files depend on the rate of Directory Server activity, the type and level of logging, and the strategy for log rotation.

Many logging requirements can be predicted and planned in advance. If Directory Server writes to logs and in particular audit logs, disk use increases with load level. When high load deployments call for extensive logging, plan for extra disk space to accommodate the high load. You may decrease disk space requirements for deployments with high load logging by establishing an intelligent log rotation and archival system, rotating the logs often, and automating migration of old files to less expensive, higher capacity storage mediums such as tape or cheaper disk clusters.

Some logging requirements cannot easily be predicted. Debug logging can cause temporary but explosive growth in the size of the `errors` log, for example. For a large, high load deployment, consider setting aside several gigabytes of dedicated disk space for temporary, high-volume debug logging. Refer to Chapter 8, “Tuning Logging,” for further information.

Transaction Log

Transaction log volume depends upon peak write loads. If writes occur in bursts, transaction logs use more space than if the write load is constant. Directory Server trims transaction logs periodically. Transaction logs therefore should not continue to grow unchecked. Transaction logs are not, however, flushed during online backup.

Directory Server is generally run with durable transactions enabled. When durable transaction capabilities are enabled, Directory Server performs a synchronous write to the transaction log for each modification (`add`, `delete`, `modify`, `modrdn`) operation. In this case, an operation can be blocked if the disk is busy, resulting in a potential I/O bottleneck.

If update performance is critical, plan to use a disk subsystem having fast write cache for the transaction log. Refer to Chapter 8, "Tuning Logging," for further information.

Replication Changelog Database

If the deployment involves replication, the Directory Server suppliers perform change logging. Changelog size depends on the volume of modifications and on the type of changelog trimming employed. Plan capacity based on how the changelog is trimmed. For a large, high load deployment, consider setting aside several gigabytes of disk space to handle changelog growth during periods of abnormally high modification rates. Refer to Chapter 8, "Tuning Logging," for further information.

Suffix Initialization and LDIF Files

During suffix initialization, also called bulk loading or importing, the Directory Server requires disk space not only for the suffix database files and the LDIF used to initialize the suffix, but also for intermediate files used during the initialization process. Plan extra (temporary) capacity in the same directory as the database files for the LDIF files and for the intermediate files used during suffix initialization.

Backups and LDIF Files

Backups often consume a great deal of disk space. The size of a backup equals the size of the database files involved. Accommodate for several backups by allocating space equal to several times the volume of the database files, ensuring that databases and their corresponding backups are maintained on separate disks. Employ intelligent strategies for migrating backups to cheaper storage mediums as they age.

If the deployment involves replication, plan additional space to hold initialization LDIF files, as these differ from backup LDIF files.

Memory Based Rather Than Disk Based File Systems

Some systems support memory based `tmpfs` file systems. On Solaris for example `/tmp` is often mounted as a memory based file system to increase performance. If cache files are placed on `/tmp`, a location shared with other applications on the system, ensure that the system never runs out of space under `/tmp`. Otherwise, when memory is low, Directory Server files in memory based file systems may be paged to the disk space dedicated for the swap partition.

Some systems support RAM disks and other alternative memory based file systems. Refer to the operating system documentation for instructions on creating and administering memory based file systems. Notice that everything in such file systems is volatile and must be reloaded into memory after system reboot.

(UNIX Platforms) Core Files

Leave room for at minimum one or two `core` files. Although Directory Server should not dump core, recovery and troubleshooting after a crash can be greatly simplified if the `core` file generated during the crash is available for inspection. When generated, `core` files are stored either in the same directory as the file specified by `nsslapd-errorlog` on `cn=config`, or under *ServerRoot*`/bin/slapd/server/` if a crash occurs during startup.

Space for Administration

Leave room for expected system use, including system and Directory Server administration. Ensure that sufficient space is allocated for the base Directory Server installation, for the configuration suffix if it resides on the local instance, for configuration files, and so forth.

Distributing Files Across Disks

By placing commonly-updated Directory Server database and log files on separate disk subsystems, you can spread I/O traffic across multiple disk spindles and controllers, avoiding I/O bottlenecks. Consider providing dedicated disk subsystems for each of the following items.

Transaction Logs

When durable transaction capabilities are enabled, Directory Server performs a synchronous write to the transaction log for each modification operation. An operation is thus blocked when the disk is busy. Placing transaction logs on a dedicated disk can improve write performance, and increase the modification rate Directory Server can handle.

Refer to “Transaction Logging,” on page 147 for more information.

Databases

Multiple database support allows each database to reside on its own physical disk. You can thus distribute the Directory Server load across multiple databases each on its own disk subsystem. To prevent I/O contention for database operations, consider placing each set of database files on a separate disk subsystem.

For top performance, place database files on a dedicated fast disk subsystem with a large I/O buffer. Directory Server reads data from the disk when it cannot find candidate entries in cache. It regularly flushes writes. Having a fast, dedicated disk subsystem for these operations can alleviate a potential I/O bottleneck.

The `nsslapd-directory` attribute on `cn=config,cn=ldbm database,cn=plugins,cn=config` specifies the disk location where Directory Server stores database files, including index files. By default, such files are located under *ServerRoot*`/slapd-`*ServerID*`/db/`.

Changing database location of course requires not only that you restart Directory Server, but also that you rebuild the database completely. Changing database location on a production server can be a major undertaking, so identify your most important database and put it on a separate disk before putting the server into production.

Log Files

Directory Server provides access, error, and audit logs featuring buffered logging capabilities. Despite buffering, writes to these log files require disk access that may contend with other I/O operations. Consider placing log files on separate disks for improved performance, capacity, and management.

Refer to Chapter 8, “Tuning Logging,” for more information.

Cache Files on Memory Based File Systems

In a `tmpfs` file system, for example, files are swapped to disk only when physical memory is exhausted. Given sufficient memory to hold all cache files in physical memory, you may derive improved performance by allocating equivalent disk space for a `tmpfs` file system on Solaris platforms or other memory based file systems such as RAM disks for other platforms, and setting the value of `nsslapd-db-home-directory` to have the Directory Server store cache files on that file system. This prevents the system from unnecessarily flushing memory mapped cache files to disk.

Disk Subsystem Alternatives

"Fast, cheap, safe: pick any two." — Sun Performance and Tuning, Cockroft and Pettit.

Fast and Safe

When implementing a deployment in which both performance and uptime are critical, consider hardware-based RAID controllers having non-volatile memory caches to provide high speed buffered I/O distributed across large arrays of disks. By spreading load across many spindles and buffering access over very fast connections, I/O can be optimized, and excellent stability provided through high performance RAID striping or parity blocks.

Large non-volatile I/O buffers and high performance disk subsystems such as those offered in Sun StorEdge™ products can greatly enhance Directory Server performance and uptime.

Fast write cache cards provide potential write performance improvements, especially when dedicated for transaction log use. Fast write cache cards provide non-volatile memory cache that is independent from the disk controller.

Fast and Cheap

For fast, low-cost performance, ensure you have adequate capacity distributed across a number of disks. Consider disks having high rotation speed and low seek times. For best results, dedicate one disk to each distributed component. Consider using multi-master replication to avoid single points of failure.

Cheap and Safe

For cheap, safe configurations, consider low-cost, software-based RAID controllers such as Solaris Volume Manager.

RAID Alternatives

RAID stands for Redundant Array of Inexpensive Disks. As the name suggests, the primary purpose of RAID is to provide resiliency. If one disk in the array fails, data on that disk is not lost but remains available on one or more other disks in the array. To implement resiliency, RAID provides an abstraction allowing multiple disk drives to be configured as a larger virtual disk, usually referred to as a volume. This is achieved by concatenating, mirroring, or striping physical disks. Concatenation is implemented by having blocks of one disk logically follow those of another disk. For example, disk 1 has blocks 0-99, disk 2 has blocks 100-199 and so forth. Mirroring is implemented by copying blocks of one disk to another and then keeping them in continuous synchronization. Striping uses algorithms to distribute virtual disk blocks over multiple physical disks.

The purpose of striping is performance. Random writes can be dealt with very quickly as data being written is likely to be destined for more than one of the disks in the striped volume, hence the disks are able to work in parallel. The same applies to random reads. For large sequential reads and writes the case may not be quite so clear. It has been observed, however, that sequential I/O performance can be improved. An application generating many I/O requests can swamp a single disk controller, for example. If the disks in the striped volume all have their own dedicated controller, however, swamping is far less likely to occur and so performance is improved.

RAID can be implemented using either a software or a hardware RAID manager device. There are advantages and disadvantages in using either method:

- Hardware RAID generally provides higher performance as it is implemented in hardware and hence incurs less processing overhead than software RAID. Furthermore, hardware RAID is dissociated from the host system, leaving host resources free to execute applications.
- Hardware RAID is generally more expensive than software RAID.
- Software RAID can be more flexible than hardware RAID. For example, a hardware RAID manager is usually associated with a single array of disks or with a prescribed set of arrays, whereas software RAID can encapsulate any number of arrays of disks, or, if desired, only certain disks within an array.

The following sections discuss RAID configurations, known as levels. The most common RAID levels, 0, 1, 1+0 and 5 are covered in some detail, whereas less common levels are merely compared and contrasted.

RAID 0, Striped Volume

Striping spreads data across multiple physical disks. The logical disk, or volume, is divided into chunks or stripes and then distributed in a round-robin fashion on physical disks. A stripe is always one or more disk blocks in size, with all stripes having the same size.

The name RAID 0 is a contradiction in that it provides no redundancy. Any disk failure in a RAID 0 stripe causes the entire logical volume to be lost. RAID 0 is, however, the least expensive of all RAID levels as all disks are dedicated to data.

RAID 1, Mirrored Volume

The purpose of mirroring is to provide redundancy. If one of the disks in the mirror fails then the data remains available and processing may continue. The trade off is that each physical disk is mirrored, meaning that half the physical disk space is devoted to mirroring.

RAID 1+0

Also known as RAID 10, RAID 1+0 provides the highest levels of performance and resiliency. Consequently, it is the most expensive level of RAID to implement. Data continues to remain available after up to three disk failures as long as all of the disks that fail form different mirrors. RAID 1+0 is implemented as a striped array where segments are RAID 1.

RAID 0+1

RAID 0+1 is slightly less resilient than RAID 1+0. A stripe is created and then mirrored. If one or more disks fails on the same side of the mirror, then the data remains available. If a disk then fails on the other side of the mirror, however, the logical volume is lost. This subtle difference with RAID 1+0 means disks on either side can fail simultaneously yet data remains available. RAID 0+1 is implemented as a mirrored array where segments are RAID 0.

RAID 5

RAID 5 is not as resilient as mirroring yet nevertheless provides redundancy in that data remains available after a single disk failure. RAID 5 implements redundancy using a parity stripe created by performing logical exclusive or on bytes of corresponding stripes on other disks. When one disk fails, data for that disk is recalculated using the data and parity in the corresponding stripes on the remaining disks. Performance suffers however when such corrective calculations must be performed.

During normal operation, RAID 5 usually offers lower performance than RAID 0, 1+0 and 0+1, as a RAID 5 volume must do four physical I/O operations for every logical write. The old data and parity are read, two exclusive or operations are performed, and the new data and parity are written. Read operations do not suffer the same penalty and thus provide only slightly lower performance than a standard stripe using an equivalent number of disks. That is, the RAID 5 volume has effectively one less disk in its stripe because the space is devoted to parity. This means a RAID 5 volume is generally cheaper than RAID 1+0 and 0+1, because RAID 5 devotes more of the available disk space to data.

Given the performance issues, RAID 5 is not generally recommended unless the data is read-only or unless there are very few writes to the volume. Disk arrays with write caches and fast exclusive or logic engines can mitigate these performance issues however, making RAID 5 a cheaper, viable alternative to mirroring for some deployments.

RAID Levels 2, 3, and 4

RAID levels 2 and 3 are good for large sequential transfers of data such as video streaming. Both levels can process only one I/O operation at time, making them inappropriate for applications demanding random access. RAID 2 is implemented using Hamming error correction coding (ECC). This means three physical disk drives are required to store ECC data, making it more expensive than RAID 5, but less expensive than RAID 1+0 as long as there are more than three disks in the stripe. RAID 3 uses a bitwise parity method to achieve redundancy. Parity is not distributed as per RAID 5, but is instead written to a single dedicated disk.

Unlike RAID levels 2 and 3, RAID 4 uses an independent access technique where multiple disk drives are accessed simultaneously. It uses parity in a manner similar to RAID 5, except parity is written to a single disk. The parity disk can therefore become a bottleneck as it is accessed for every write, effectively serializing multiple writes.

Software Volume Managers

Volume managers such as Solaris™ Volume Manager may also be used for Directory Server disk management. Solaris Volume Manager compares favorably with other software volume managers for deployment in production environments.

Monitoring I/O and Disk Use

Disks should not be saturated under normal operating circumstances. You may use utilities such as `iostat`(1M) on Solaris and other systems to isolate potential I/O bottlenecks. Refer to Windows help for details on handling I/O bottlenecks on Windows systems.

Sizing for Multiprocessor Systems

Directory Server software is optimized to scale across multiple processors. In general, adding processors may increase overall search, index maintenance, and replication performance.

In specific directory deployments, however, you may reach a point of diminishing returns where adding more processors does not impact performance significantly. When handling extremely demanding performance requirements for searching, indexing, and replication, consider load balancing and directory proxy technologies as part of the solution.

Sizing Network Capacity

Directory Server is a network intensive application. To improve network availability for a Directory Server instance, equip the system with two or more network interfaces. Directory Server can support such a hardware configuration, listening on multiple network interfaces within the same process.

If you intend to cluster directory servers on the same network for load balancing purposes, ensure the network infrastructure can support the additional load generated. If you intend to support high update rates for replication in a wide area network environment, ensure through empirical testing that the network quality and bandwidth meet your requirements for replication throughput.

Sizing for SSL

By default, support for the Secure Sockets Layer (SSL) protocol is implemented in software. Using the software-based SSL implementation may have significant negative impact on Directory Server performance. Running the directory in SSL mode may require the deployment of several directory replicas to meet overall performance requirements.

Although hardware accelerator cards cannot eliminate the impact of using SSL, they can improve performance significantly compared with software-based implementation. Sun ONE Directory Server 5.2 supports the use of SSL hardware accelerators such as supported Sun Crypto Accelerator hardware.

Using a Sun Crypto Accelerator board can be useful when SSL key calculation is a bottleneck. Such hardware may not improve performance when SSL key calculation is not a bottleneck, however, as it specifically accelerates key calculations during the SSL handshake to negotiate the connection, but not encryption and decryption of messages thereafter. Refer to Appendix B, "Using the Sun Crypto Accelerator Board," for instructions on using such hardware with a Directory Server instance.

Chapter 5

Tuning the Operating System

Default system and network settings are not suitable for high performance directory services. Tuning the system for optimum Directory Server performance involves at minimum checking that the latest recommended patches are installed on the system, enforcing basic security measures, and changing some system and network settings. This chapter addresses those tuning issues.

The `idsktune` utility (also `/usr/sbin/directoryserver idsktune` in the Solaris packaged version) provided with the product may help you to diagnose basic system configuration shortcomings. The utility offers system tuning recommendations for support of high performance directory services. The utility does not actually implement any of the recommendations made. Tuning recommendations should be implemented by a qualified system administrator.

Checking Platform Support

Table 1-1 on page 18 specifies platforms and associated hardware architectures supported for this release. Refer to the product release notes for an updated list of supported platforms.

When installing a Windows system, specify that the computer is a stand-alone server, not a member of any existing domain or workgroup, to reduce dependencies on network security services.

Patching the System

In order to maintain overall system security, and to ensure proper installation and operation of Sun ONE Directory Server 5.2, install the latest recommended system patches, service packs, or fixes. Table 5-1 suggests where to look for required patches.

Table 5-1 Where to Obtain Patches, By Platform

Platform	Browse...
Sun Solaris™ Operating Environment	http://sunsolve.sun.com/
Hewlett Packard HP-UX	http://www.hp.com/support/
IBM AIX	http://www.ibm.com/support/
Microsoft Windows	http://support.microsoft.com/
Red Hat Linux	http://www.redhat.com/

Enforcing Basic Security

The recommendations in this section do not eliminate all risk. Instead, they are intended as a short checklist to help you work to limit some of the most obvious security risks.

Isolate the System

If at all possible, isolate the system running Directory Server from the public Internet using a network firewall. Isolating the system is particularly important when running Directory Server on Windows platforms that must be protected from IP-based attacks.

No Dual Boot

Do not dual boot or run other operating systems on the system running Directory Server. Other systems may permit access to otherwise restricted files.

Strong Passwords

Use a super user or Administrator password at least 8 characters long that includes punctuation or other non-alphabetic characters. Using a strong password is particularly important when running Directory Server on Windows platforms.

If you choose to use longer operating system passwords, it may be necessary to configure the way passwords are handled by the system. Refer to the operating system documentation for instructions.

(Windows) Local Security Policy

Implement a local security policy for the Windows server that locks users out after bad logon attempts. Activate and configure event logging to manage a log of appropriate size for the deployment. Also activate audit logging for logon attempts. Consider renaming the Administrator account to make it harder to guess.

Refer to Windows help for details.

(UNIX Platforms) Users and Groups

For security reasons, it is recommended not to run Directory Server or Administration Server with super user privileges. You may, for example, create a user and group without login privileges, and then install and run the servers as this user and group. If you add the user and group to local files the `/etc/passwd` entry could be, for example:

```
server:x:61001:Server User:/dev/null:/dev/null
```

The corresponding `/etc/group` entry could be, for example:

```
servers::61001:
```

To facilitate debugging, you may choose to allow processes running with this user and group identity to dump core, using utilities such as `coreadm`(1M) on Solaris systems.

If a particular deployment calls for sharing Directory Server files with other servers such as a messaging server, consider running those servers using the same user and group.

If you must run the Administration Server as super user, consider stopping the service when not using it.

Disabling Unnecessary Services

For top performance and less risk, dedicate the system to Directory Server alone. Running additional services, especially network services, negatively affects server performance and scalability, and may increase security risks.

Disable as many network services as possible. Directory Server uses only TCP/IP and does not require file sharing and other services. Disable services such as IP Routing, Mail, NetBIOS, NFS, RAS, Web Publishing, and Windows Network Client services. On Windows in particular, stop and disable all services except for Event Log, Plug and Play, Protected Storage, Security Accounts Manager, Sun ONE Administration Server, Sun ONE Directory Server, Remote Procedure Call (RPC), and SNMP. Consider disabling `telnet` and `ftp`.

As with many network services, `telnet` and `ftp` pose security risks. These two services are particularly dangerous in that they transmit user passwords in clear text over the network. You may be able to work around the need `telnet` and `ftp` by using clients such as Secure Shell (`ssh`) and Secure FTP (`sftp`) instead.

If the Directory Server instance does not itself provide the naming service for the network, consider enabling a naming service for the system. Remote administration tools such as Sun ONE Server Console rely on the naming service for some aspects of their operation such as translating between IP addresses and host names.

Refer to the operating system documentation for details on disabling network services.

Keeping Accurate Time

Ensure the system clock is reasonably in sync with those of other systems to facilitate replication and correlation of date and time stamps in log files between systems. Consider using a Network Time Protocol (NTP) client to set the correct system time, for example, especially on Windows systems.

Restarting After System Failure

When possible, stop Directory Server as described in the *Sun ONE Directory Server Administration Guide*. Database corruption may cause Directory Server to start slowly if stopped abruptly during system shutdown, rather than shut down appropriately. Time may be needed to recover the database.

(Solaris Packages) As part of the installation and configuration process, appropriate scripts enable restart at boot time.

(Windows) Configure Windows to restart automatically after system failure. Refer to Windows help for details.

For other platforms, refer to the operating system documentation for details on starting services at boot time.

Generating Basic Tuning Recommendations

Use the `idsktune` utility, `/usr/sbin/directoryserver idsktune` for the Solaris packaged version, or `idsktune` for other versions located in the directory containing the product binaries, to generate basic tuning recommendations on platforms other than Windows.

When you run the utility as super user, it gathers information about information about the system. It displays notices, warnings, and errors with recommended corrective actions. For example, the utility checks that:

- Operating system and kernel versions are supported for this release.
- Available memory and disk space meet minimum requirements for typical use.
- System resource limits meet minimum requirements for typical use.
- Required patches or service packs are installed.

NOTE Fix at minimum all `ERROR` conditions before installing Directory Server software on a system intended for production use.

Individual deployment requirements may exceed minimum requirements. You may opt to provide more resources than those identified as minimum system requirements by the `idsktune` utility.

Refer to the Sun ONE Directory Server Resource Kit documentation for details concerning the utility. The Sun ONE Directory Server Resource Kit can be obtained as described in “Downloading Directory Server Tools,” on page 12.

Tuning System Settings

You may use the `idsktune` tool that reads current system settings and recommends changes. In general, implementing the recommendations optimizes performance both on systems dedicated to running Directory Server and on systems running additional applications.

Consider local network conditions and other applications before implementing specific recommendations. Refer to the operating system documentation for additional network tuning tips.

Table 5-2 Configuration Files to Check Prior to Deployment

Platform	File	Remarks
Solaris Operating Environment	`/etc/init.d/inetinit`	Add `ndd` statements for tuning
	`/etc/system`	Check system limits
	`/etc/vfstab`	Ensure files are local
HP-UX	`/etc/rc.config.d/nddconf`	Add `ndd` statements for tuning
		Alternatively, use `sam`(1M).
Red Hat Linux	`/etc/fstab`	Ensure files are local
	`/etc/security/limits.conf`	Add `nofile` hard limit directive
	`/etc/sysctl.conf`	Check, set kernel parameters
	`/proc/sys/fs/file-max`	Check file descriptor limits

(Windows) Deferred Procedure Calls

Windows default Deferred Procedure Call (DPC) handling of deferred interrupt requests to handle incoming network traffic in multiprocessor systems can have negative performance impact. In effect, deferred interrupts that become DPCs may be rescheduled from one processor to another, incurring significant overhead. To prevent these DPC overhead problems, set the value of `ProcessorAffinityMask` to `0` under the following registry key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\NDIS\Parameters
```

File Descriptors

Directory Server uses file descriptors when handling concurrent client connections. Having a low maximum number of file descriptors available in the system or available to a process can thus limit the number of concurrent connections. Recommendations concerning the number of file descriptors therefore relate to the number of concurrent connections Directory Server may be able to handle on the system.

On Solaris systems, the number of file descriptors available is configured through the `rlim_fd_max` parameter, as described in the output of `/usr/sbin/directoryserver idsktune` (packaged version) or `idsktune` (no packages). Refer to the operating system documentation for further instructions on modifying the number of available file descriptors.

After modifying the maximum number of available file descriptors on the system, refer to Table 9-2 on page 152 for information on configuring Directory Server to use the available file descriptors.

(HP-UX) Large File Support

On some HP-UX systems, large files are not supported by default. Refer to the HP-UX product documentation for instructions on enabling large file support on the file systems where you intend to install Directory Server. Instructions are also provided in the output of the `idsktune` utility.

(HP-UX) Threads Pending Timeout

On some HP-UX systems, the maximum number of threads pending timeouts may be inappropriately set as indicated by the `idsktune` utility. Refer to the HP-UX documentation for instructions on increasing the maximum number of threads pending timeouts. Instructions are also provided in the output of the `idsktune` utility.

(HP-UX) Threads Per Process

On some HP-UX systems, the maximum number of threads per process may be too low as indicated by the `idsktune` utility. Refer to the HP-UX documentation for instructions on increasing the maximum number of threads per process. Instructions are also provided in the output of the `idsktune` utility.

Transmission Control Protocol (TCP) Settings

Specific network settings depend on the platform. On some systems, it is possible to enhance Directory Server performance by modifying TCP settings. This section discusses the reasoning behind `idsktune` recommendations concerning TCP settings.

Closed Connections in the TIME-WAIT State

Some systems allow you to configure how long a TCP connection is held in the kernel table after closure. While the connection is held, it may be opened again quickly. When set too high, the system may track a large number of connections in the kernel table over long intervals, reducing the number of available connections to Directory Server. For most deployments, setting this parameter to a value of 30 seconds (30,000 milliseconds) allows more concurrent connections to Directory Server.

On Solaris systems, this time interval is configured through the `tcp_time_wait_interval` parameter, as described in the output of `/usr/sbin/directoryserver idsktune` (packaged version) or `idsktune` (no packages).

Connections Pending Acceptance

Some systems allow you to configure the number of TCP connections pending acceptance by a TCP listener such as Directory Server. When set too low, this limits the number of pending connections Directory Server can accept. For most deployments, setting this parameter to a value of at least 1024 allows Directory Server to handle more concurrent connection requests.

On Solaris systems, the number of pending connections allowed is configured through the `tcp_conn_req_max_q` parameter, as described in the output of `/usr/sbin/directoryserver idsktune` (packaged version) or `idsktune` (no packages). Consider increasing `tcp_conn_req_max_q0` to 2048.

Deferred Acknowledgement

Some systems allow you to configure how long TCP acknowledgements are delayed for hosts not directly connected to the system. Instead of configuring delay times directly, set `nsslapd-nagle` on `cn=config` to `off` as described in Table 9-2 on page 152.

Inactive Connections

Some systems allow you to configure the interval between transmission of keepalive packets. This setting can determine how long a TCP connection is maintained while inactive and potentially disconnected. When set too high, the keepalive interval may cause the system to use unnecessary resources keeping connections alive for clients that have become disconnected. For most deployments, setting this parameter to a value of 600 seconds (600,000 milliseconds = 10 minutes) allows more concurrent connections to Directory Server.

On Solaris systems, this time interval is configured through the `tcp_keepalive_interval` parameter, as described in the output of `/usr/sbin/directoryserver idsktune` (packaged version) or `idsktune` (no packages).

Incoming Connections

Some systems allow you to configure how long a system waits for an incoming connection not sending acknowledgements. When set too high, this can cause long delays in detecting connection failure. For intranet deployments on fast and reliable networks, setting this parameter to a value of 600 seconds (600,000 milliseconds = 10 minutes) may improve performance.

On Solaris systems, this time interval is configured through the `tcp_ip_abort_interval` parameter, as described in the output of `/usr/sbin/directoryserver idsktune` (packaged version) or `idsktune` (no packages).

Outgoing Connections

Some systems allow you to configure how long a system waits for an outgoing connection to be established. When set too high, establishing outgoing connections to destination servers such as replicas not responding quickly can cause long delays. For intranet deployments on fast and reliable networks, setting this parameter to a value of 10 seconds may improve performance.

On Solaris systems, this time interval is configured through the `tcp_ip_abort_cinterval` parameter, as described in the output of `/usr/sbin/directoryserver idsktune` (packaged version) or `idsktune` (no packages).

Retransmission Timeout

Some systems allow you to configure the initial time interval between retransmission of packets. This setting affects the wait before retransmission of an unacknowledged packet. When set too high, clients may be kept waiting on lost packets. For intranet deployments on fast and reliable networks, setting this parameter to a value of 500 milliseconds may improve performance.

On Solaris systems, this time interval is configured through the `tcp_rexmit_interval_initial` parameter, as described in the output of `/usr/sbin/directoryserver idsktune` (packaged version) or `idsktune` (no packages).

Windows can implement the Van Jacobson TCP fast retransmit and recovery algorithm to retransmit quickly missing segments upon the receipt of an ACK without waiting for the retransmission timer to expire. To implement the Van Jacobson algorithm, edit the registry key:

```
HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/Tcpip/Parameters
```

Add `TcpMaxDupAcks` having type `REG_DWORD`. Set the value to the number of ACKs. The range is 1-3, and the default is 2.

Sequence Numbers

Some systems allow you to configure how the system handles initial sequence number generation. For extranet and Internet deployments, set this parameter such that initial sequence number generation is based on RFC 1948 to prevent sequence number attacks.

On Solaris systems, this behavior is configured through the `tcp_strong_iss` parameter, as described in the output of `/usr/sbin/directoryserver idsktune` (packaged version) or `idsktune` (no packages).

Chapter 6

Tuning Cache Sizes

Directory Server caches directory information in memory and on disk in order to be able to respond more quickly to client requests. Properly tuned caching minimizes the need to access disk subsystems when handling client requests.

NOTE Unless caches are tuned and working properly, other tuning may have only limited impact on performance.

Types of Cache

Directory Server handles three types of cache as described in Table 6-1.

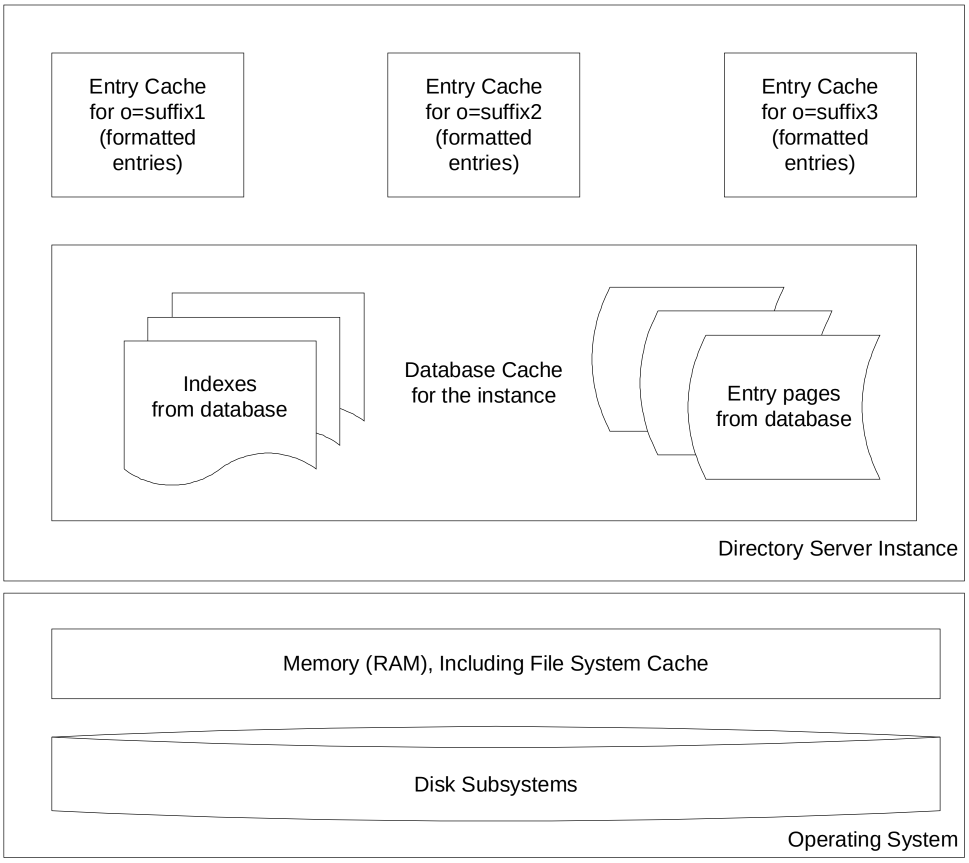
Table 6-1 Caches

Cache Type	Description
Database	Each Directory Server instance has one database cache that holds both indexes and entries in database format.
Entry	Each suffix has an entry cache that holds entries retrieved from the database during previous operations and formatted for quick delivery to client applications.
Import	Each Directory Server instance has an import cache that is structurally similar to the database cache and is used during bulk loading.

Directory Server also benefits from file system cache, handled by the underlying operating system, and from I/O buffers in disk subsystems.

Figure 6-1 shows caches for an instance of Directory Server handling three suffixes, each with its own entry cache. The instance is configured to handle significant disk activity, placing the transaction log, database, and other files and logs on separate disk subsystems as suggested in Chapter 8, "Tuning Logging."

Figure 6-1 Entry and Database Caches in Context


Entry Cache for o=suffix1 (formatted entries)
Entry Cache for o=suffix2 (formatted entries)
Entry Cache for o=suffix3 (formatted entries)
Indexes from database
Database Cache for the instance
Entry pages from database
Directory Server Instance
Memory (RAM), Including File System Cache
Disk Subsystems
Operating System

Database Cache

Each Directory Server instance has one database cache. The database cache holds pages from the database containing indexes and entries. Each page is not an entry, but a slice of memory containing a portion of the database. You specify database cache size (`nsslapd-dbcachesize`). The change to database cache size takes effect after you restart the server, with database cache space allocated at server startup.

Directory Server moves pages between the database files and the database cache to maintain maximum database cache size. The actual amount of memory used by Directory Server for database cache may be up to 25 percent larger than the size you specify, due to additional memory needed to manage the database cache itself.

When using a very large database cache, verify through empirical testing and by monitoring memory use with tools such as `pmap`(1) on Solaris systems that the memory used by Directory Server does not exceed the size of available physical memory. Exceeding available physical memory causes the system to start paging repeatedly, resulting in severe performance degradation.

The `ps`(1) utility, present on the UNIX platforms Directory Server supports, can also be used with the `-p` *pid* and `-o` *format* options to view current memory used by a particular process such as Directory Server (`ns-slapd`). On Windows systems, the Task Manager Processes tab page lists memory usage per process (`slapd.exe`). Refer to the operating system documentation for details.

For 32-bit servers, database cache size must in practice be limited to 2 GB or less.

NOTE	On Windows and AIX platforms, do not allocate more than 1 GB (1,073,741,824 bytes) for database cache.

Refer to the *Sun ONE Directory Server Reference Manual* for further details concerning the valid range of `nsslapd-dbcachesize` values.

Entry Cache

The entry cache holds recently accessed entries, formatted for delivery to client applications. You specify entry cache size for a suffix (`nsslapd-cachememsize`) and the maximum number of entries (`nsslapd-cachesize`). Entry cache is allocated as needed.

Directory Server can return entries from an entry cache extremely efficiently, as entries stored in this cache are already formatted. Entries in the database are stored as a raw string of bytes, and must be formatted (and stored in the entry cache) before delivery to client applications.

When specifying entry cache size, know that `nsslapd-cachememsize` indicates how much memory Directory Server requests from the underlying memory allocation library. Depending on how the memory allocation library handles such requests, actual memory used may be much larger than the effective amount of memory ultimately available to Directory Server for the entry cache.

Actual memory used by the Directory Server process depends primarily on the memory allocation library used, and on the entries cached. Entries with many small attribute values usually require more overhead than entries with a few large attribute values.

For 32-bit servers, entry cache size must in practice be limited to 2 GB or less.

NOTE On AIX platforms, Directory Server is created with `maxdata = 0x50000000`, allowing you to allocate 1 GB each to both database cache and entry cache. Contact your Sun ONE support representative if you must change the value of `maxdata`.

Refer to the *Sun ONE Directory Server Reference Manual* for further details concerning the valid range of `nsslapd-cachememsize` and `nsslapd-cachesize` values.

Import Cache

The import cache is created and used during suffix initialization only, also known as bulk loading or importing. If the deployment involves *offline* suffix initialization only, import cache and database cache are not used together, so you need not add them together when aggregating cache size as described in "Total Aggregate Cache Size." You specify import cache size (`nsslapd-import-cachesize`). Changes to import cache size take effect the next time the suffix is reset and initialized, with import cache allocated for the initialization, then released after the initialization.

Directory Server handles import cache as it handles database cache. Ensure therefore that sufficient physical memory is available to prevent swapping.

For 32-bit servers, import cache size must in practice be limited to 2 GB or less. Refer to the *Sun ONE Directory Server Reference Manual* for further details concerning the valid range of `nsslapd-import-cachesize` values.

File System Cache

The operating system allocates available memory not used by Directory Server caches and other applications to the file system cache. This cache holds data recently read from the disk, making it possible for subsequent requests to obtain data copied from cache rather than to read it again from the disk. As memory access is many times faster than disk access, leaving some physical memory available to the file system cache can boost performance.

Refer to the operating system documentation for details on file system cache.

Total Aggregate Cache Size

The sum of all caches used simultaneously must remain smaller than the total size of available physical memory, less the memory intended for file system cache. For 32-bit servers, this means total aggregate cache size must in practice be limited to 2 GB or less. *Total cache used may well be significantly larger than the size you specify.* Refer to "Database Cache," on page 108 for hints on how to check that the cache size and thus Directory Server process size does not exceed available physical memory.

NOTE On Windows platforms, the maximum address space available to an application is 2 GB. If the total aggregate cache size exceeds this limit, Directory Server exits with an error message.

If suffixes are initialized (bulk loaded) while Directory Server is online, the sum of database, entry, and import cache sizes should remain smaller than the total size of available physical memory.

Table 6-2 Suffix Initialization (Import) Operations and Cache Use

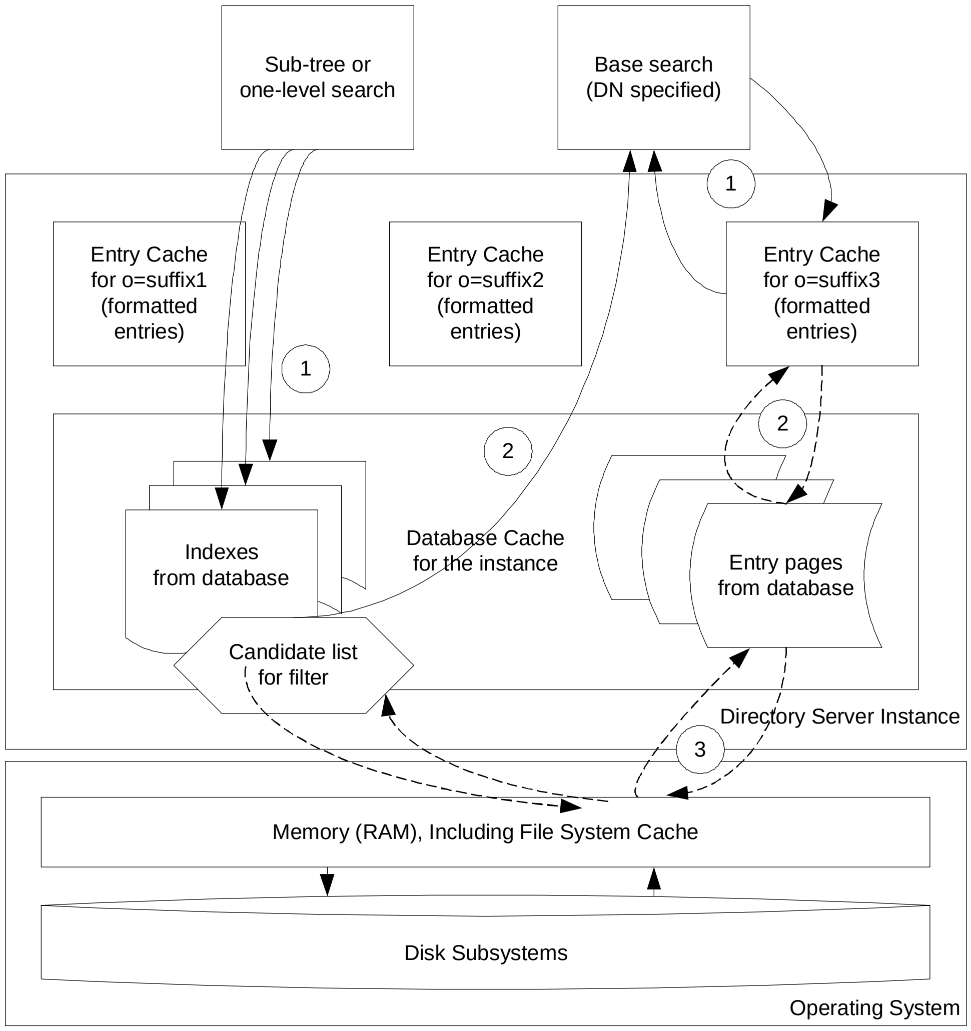
Cache Type	Offline Import	Online Import
Database	no	yes
Entry	yes	yes
Import	yes	yes

If all suffix initialization takes place offline with Directory Server stopped, you may be able to work around this limitation. In this case import cache does not coexist with database cache, so you may allocate the same memory to import cache for offline suffix initialization and to database cache for online use. If you opt to implement this special case, however, ensure that no one performs online bulk loads on the production system. The sum of the caches used simultaneously must still remain smaller than the total size of available physical memory.

How Searches Use Cache

Figure 6-2 illustrates how Directory Server handles both searches specifying a base DN and searches using filters. Individual lines represent threads accessing different levels of memory, with broken lines representing steps to minimize through effective tuning.

Figure 6-2 Searches and Cache


Sub-tree or one-level search
Base search (DN specified)
1
Entry Cache for o=suffix1 (formatted entries)
Entry Cache for o=suffix2 (formatted entries)
Entry Cache for o=suffix3 (formatted entries)
1
2
2
Indexes from database
Database Cache for the instance
Entry pages from database
Candidate list for filter
Directory Server Instance
3
Memory (RAM), Including File System Cache
Disk Subsystems
Operating System

Base Search Process

As shown, base searches (those specifying a base DN) are the simplest type of searches for Directory Server to handle. To process such searches, Directory Server:

1. Attempts to retrieve the entry having the specified base DN from the entry cache.

   If the entry is found there, Directory Server checks whether the candidate entry matches the filter provided for the search.

   If the entry matches, Directory Server then quickly returns the formatted, cached entry to the client application.

2. Attempts to retrieve the entry from the database cache.

   If the entry is found there, Directory Server copies the entry to the entry cache for the suffix, and then proceeds as if the entry had been found in the entry cache.

3. Attempts to retrieve the entry from the database itself.

   If the entry is found there, Directory Server copies the entry to the database cache, then proceeds as if the entry had been found in the database cache.

Subtree and One-Level Search Process

Also as shown in Figure 6-2 on page 113, searches on a subtree or a level of a tree involve additional processing to handle sets of entries. To process such searches, Directory Server:

1. Attempts to build a set of candidate entries that match the filter from indexes in the database cache.

   If no appropriate index is present, the set of candidate entries must be generated from the relevant entries in the database itself.

2. Handles each candidate entry by:

   a. Performing a base search to retrieve the entry.

   b. Checking whether the entry matches the filter provided for the search.

   c. Returning the entry to the client application if the entry matches the filter.

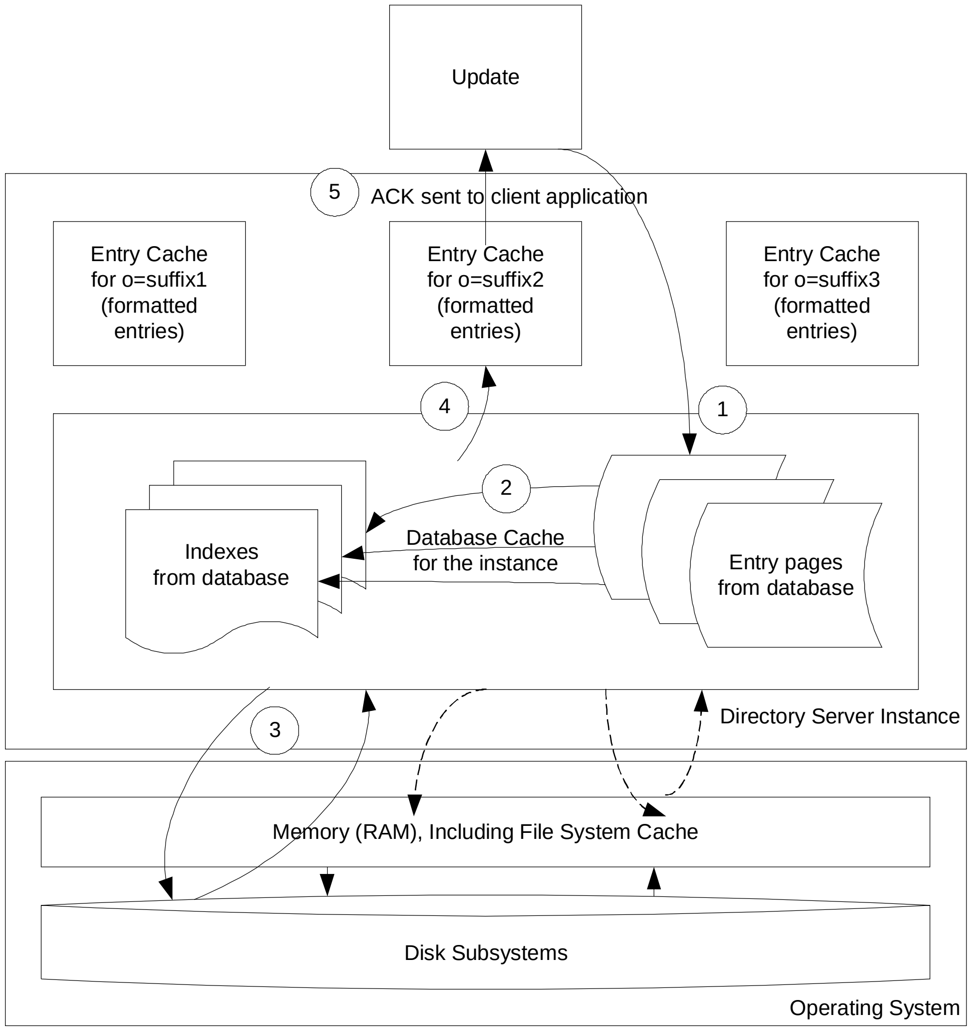
   In this way, Directory Server avoids constructing the set in memory.

Ideally, you know what searches to expect before tuning Directory Server. In practice, verify assumptions through empirical testing.

How Updates Use Cache

Figure 6-3 illustrates how Directory Server handles updates. Individual lines represent threads accessing different levels of memory, with broken lines representing steps to minimize through effective tuning.

Figure 6-3 Updates and Cache


Update
5
ACK sent to client application
Entry Cache for o=suffix1 (formatted entries)
Entry Cache for o=suffix2 (formatted entries)
Entry Cache for o=suffix3 (formatted entries)
4
1
2
Indexes from database
Database Cache for the instance
Entry pages from database
3
Directory Server Instance
Memory (RAM), Including File System Cache
Disk Subsystems
Operating System

Updates involve more processing than searches. To process updates, Directory Server:

1. Performs a base DN search to retrieve the entry to update or verify in the case of an add operation that it does not already exist.
2. Changes the database cache, updating in particular any indexes affected by the update.

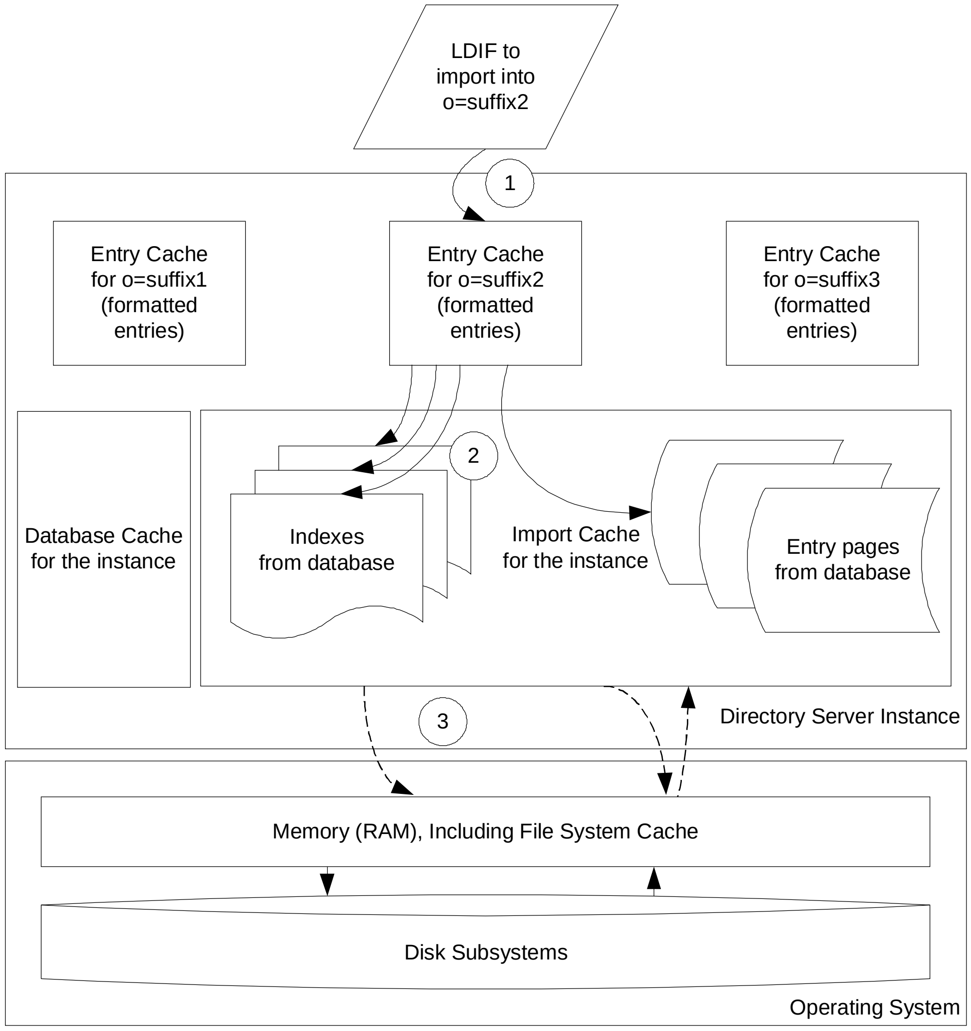
   If the data affected by the update has not been loaded into the database cache, this step can result in disk activity while the relevant data are loaded into the cache.
3. Writes information about the changes to the transaction log, waiting for the information to be flushed to disk.

   Refer to "Transaction Logging," on page 147 for details.
4. Formats and copies the updated entry to the entry cache for the suffix.
5. Returns an acknowledgement of successful update to the client application.

How Suffix Initialization Uses Cache

Figure 6-4 illustrates how Directory Server handles suffix initialization, also known as bulk load import. Individual lines represent threads accessing different levels of memory, with broken lines representing steps to minimize through effective tuning.

Figure 6-4 Suffix Initialization (Bulk Loading) and Cache


LDIF to import into o=suffix2
1
Entry Cache for o=suffix1 (formatted entries)
Entry Cache for o=suffix2 (formatted entries)
Entry Cache for o=suffix3 (formatted entries)
2
Database Cache for the instance
Indexes from database
Import Cache for the instance
Entry pages from database
3
Directory Server Instance
Memory (RAM), Including File System Cache
Disk Subsystems
Operating System

To initialize a suffix, Directory Server:

1. Starts a thread to feed an entry cache, used as a buffer, from LDIF.
2. Starts a thread for each index affected and a thread to create entries in the import cache. These threads consume entries fed into the entry cache.

3. Reads from and writes to the database files when import cache runs out.

   Directory Server may also write log messages during suffix initialization, but does not write to the transaction log.

Tools for suffix initialization such as `ldif2db` (`/usr/sbin/directoryserver ldif2db`) delivered with Directory Server provide feedback concerning cache hit rate and import throughput. Having both cache hit rate and import throughput drop together suggests that import cache may be too small. Consider increasing import cache size.

Optimizing For Searches

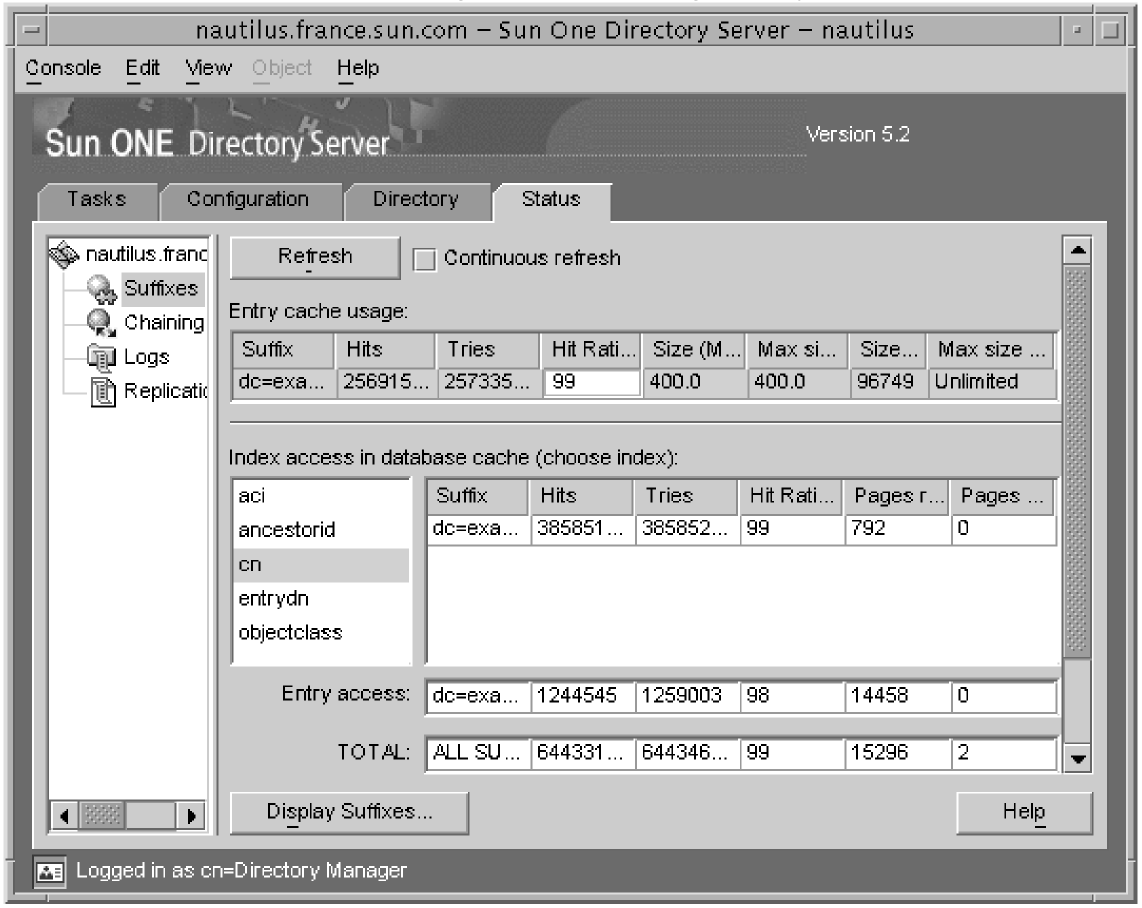
For top performance, cache as much directory data as possible in memory. In preventing the directory from reading information from disk, you limit the disk I/O bottleneck. There are a number of different possibilities for doing this, depending on the size of your directory tree, the amount of memory available and the hardware used. Depending on the deployment, you may choose to allocate more or less memory to entry and database caches to optimize search performance. You may alternatively choose to distribute searches across Directory Server consumers on different servers.

All Entries and Indexes in Memory

Imagine the optimum case. Database and entry caches fit into the physical memory available. The entry cache is large enough to hold all entries in the directory. The database cache is large enough to hold at minimum all indexes. In this case, searches find everything in cache. Directory Server never has to go to file system cache or to disk to retrieve entries.

In this case, ensure that database cache can contain all database indexes even after updates. When space runs out in the database cache for indexes, Directory Server must read indexes from disk for every search request, severely impacting throughput. Directory Server Console displays hit ratios and other useful information under the Status tab as shown in Figure 6-5.

Figure 6-5 Monitoring Cache Hit Rate Using Directory Server Console



Entry cache usage:

Suffix	Hits	Tries	Hit Rati...	Size (M...	Max si...	Size...	Max size ...
dc=exa...	256915...	257335...	99	400.0	400.0	96749	Unlimited

Index access in database cache (choose index):

aci
ancestorid
cn
entrydn
objectclass

	Suffix	Hits	Tries	Hit Rati...	Pages r...	Pages ...
	dc=exa...	385851...	385852...	99	792	0
Entry access:	dc=exa...	1244545	1259003	98	14458	0
TOTAL:	ALL SU...	644331...	644346...	99	15296	2

Alternatively, paging and cache activity can be monitored by searching from the command line:

```
$ ldapsearch -D admin -w password \
-b cn=monitor,cn=database_name,cn=ldbm database,cn=plugins,cn=config
```

As a rough estimate of the amount of memory needed for all the database indexes held in `.db3` files to fit in the database cache, use the following formula. This formula is approximately accurate for default index configuration used with typical entries having no large binary attributes such as photos.

$\texttt{nsslapd-dbcachesize = 1.2 x SUM}_{\texttt{all .db3 files}}\texttt{(file size)}$

As a rough estimate of the number of entry cache slots and the amount of memory needed for all entries to fit in the entry cache, use the following formulas. Again these formulas are approximately accurate for default index configuration used with typical entries having no large binary attributes such as photos.

```
nsslapd-cachesize = 4.5 x (number of entries in LDIF)
```

```
nsslapd-cachememsize = 3.8 x (id2entry.db3 file size)
```

Verify and correct estimates through empirical testing. Entry caches in particular may use *much more* memory than you allocate to them.

Plenty of Memory, 32-Bit Directory Server

Imagine a system with sufficient memory to hold all data in entry and database caches, but no support for a 64-bit Directory Server process. If hardware constraints prevent you from deploying on a Solaris system, for example, the key is to size caches appropriately with respect to memory limitations for 32-bit processes, then to leave available memory to the file system cache.

NOTE File system cache is shared with other processes on the system, especially file based operations. It is thus considerably more difficult to control than other caches, particularly on systems not dedicated to Directory Server.

The system may reallocate file system cache to other processes.

Less Memory, Some File System Cache

Imagine a system with insufficient available memory to hold all data in entry and database caches, but yet with still significant available memory. The key in this case is to avoid causing combined entry and database cache sizes to exceed the available physical memory, resulting in heavy virtual memory paging that could bring the system to a virtual halt.

Consider leaving available memory to the file system cache, setting entry cache and database cache sizes to low figures such as 500 KB. By doing this, you may permit the system to retain enough data in the file system cache to end searches there, before Directory Server must repeatedly read entries and indexes from the disk.

Alternatively, if search patterns are less random, you may choose to set entry and database caches higher on the assumption that most of the searches in the particular deployment access the same small subset of all entries in the directory, and that the gains from having entries and indexes cached for such searches offset the cost of handling occasional unusual search requests. Verify and correct assumptions through empirical testing.

Low Memory, Low File System Cache

Imagine a system with insufficient available memory both for holding data in entry and database caches, but also for allowing the system to cache data in the file system cache. The key in this case is to make the most of the available memory you have.

Consider setting entry and database cache sizes as low as possible, leaving as much memory as possible to the file system cache. Leaving memory to the file system cache at least prevents entries from being expanded into database or entry by a factor of 3 to 4.5, theoretically limiting the disk I/O activity. Verify this assumption through empirical testing for the particular deployment.

Optimizing for Updates

For top update performance, first remove any transaction log bottlenecks observed. Refer to “Transaction Logging,” on page 147 for details.

Next, attempt to provide enough memory for the database cache to handle updates in memory and minimize disk activity. You can monitor the effectiveness of the database cache by reading the hit ratio in Directory Server Console. Directory Server Console displays hit ratios for suffixes under the Status tab as shown in Figure 6-5 on page 119.

Attempt as well to leave significant memory available for the file system cache. After Directory Server has run for some time, the file system cache should contain enough entries and indexes that disk reads are no longer necessary. Updates should affect the database cache in memory, with data from the large database cache in memory being flushed only infrequently.

Flushing data to disk can itself be a bottleneck, so storing the database on a separate RAID system such as a Sun StorEdge™ disk array can help improve update performance. You may use utilities such as `iostat`(1M) on Solaris systems to isolate potential I/O bottlenecks. Refer to Windows help for details on handling I/O bottlenecks on Windows systems.

Cache Priming and Monitoring

Priming caches means filling them with data such that subsequent Directory Server behavior reflects normal operational performance, rather than ramp up. Prime caches before measuring and analyzing potential optimizations.

Prime the entry cache for the suffix using the `ldapsearch` utility. For example:

```
$ ldapsearch -D directoryManager -w password -b suffix objectclass=\* > /dev/null
```

Perform searches to prime the database cache and in particular to load indexes into the cache. You can prime presence indexes by performing searches with filters such as `(mail=*)`. For other indexes, consider using the Sun ONE Directory Server Resource Kit `searchrate` utility applying filter formats to search for all possible values of each attribute to index. In other words, to check performance for equality searches against `mail` attributes, for example, generate a file with one mail address per line for each mail address, then use the `searchrate` utility to perform searches using the file. For example:

```
$ searchrate -b suffix -f "(mail=%s)" -i mail.file -K -t 10
```

Consider using `-K` and `-t` to save time. When used with the `-K` option, `searchrate` holds the connection open, binding only once, not binding for each search. The `-t` option lets you specify how many threads to use. Refer to the Sun ONE Directory Server Resource Kit documentation for details concerning the `searchrate` utility. The Sun ONE Directory Server Resource Kit can be obtained as described in “Downloading Directory Server Tools,” on page 12.

After other caches are primed, you can prime available file system cache. Although you cannot guarantee information in the file system cache is not flushed, priming file system cache nevertheless may improve ramp up time. To prime files system cache on UNIX systems you may use the `dd`(1M) command as super user. On Solaris systems with database files in the default location for example:

```
# for db in ServerRoot/slapd-serverID/db/*/*.db3
> do
> dd if='pwd'/$db of=/dev/null bs=512k
> done
0+1 records in
0+1 records out
...
```

After caches are primed, you may run tests, and monitor whether cache tuning has produced the desired outcomes. Directory Server Console displays monitoring information for caches when you select the Suffixes node under the Status tab as shown in Figure 6-5 on page 119. Alternatively, paging and cache activity can be monitored by searching from the command line:

```
$ ldapsearch -D admin -w password \
-b cn=monitor,cn=database_name,cn=ldbm\ database,cn=plugins,cn=config
```

If database cache size is large enough and the cache is primed, then the hit ratio (`dbcachehitratio`) should be high, and number of pages read in (`dbcachepagein`) and clean pages written out (`dbcacheroevict`) should be low. Here, "high" and "low" must be understood relative to the deployment constraints.

If entry cache for a suffix is large enough and the cache is primed, then the hit ratio (`entrycachehitratio`) should be high. The entry cache size (`currententrycachesize`) should be no more than 80 percent of the maximum size (`maxentrycachesize`). Finally, the size in entries (`currententrycachecount`) should be either equal or very close to the total number of entries in the suffix.

Other Optimizations

Tuning cache sizes represent only one approach to improving search, update or bulk load rates. As you tune the cache, performance bottlenecks from cache move to other parts of the system. Refer to the other chapters in this guide for more information.

Chapter 7

Tuning Indexing

As Directory Server handles more and more entries, searches potentially consume more and more time and system resources. Indexes are one tool to improve search performance. This chapter covers how Directory Server indexes work so that you understand the costs and benefits of using a specific index in the context of a particular deployment.

About Indexes

Indexes associate lookup information with Directory Server entries. Indexes take the form of files stored with Directory Server databases. A *database* in this context is the physical representation of a suffix. For most deployments, one suffix corresponds to one database. For some deployments, one suffix may be split across multiple databases. Directory Server stores databases under *ServerRoot*`/slapd-`*ServerID*`/db/` by default (the default value of `nsslapd-directory`). Here you find individual database instances having one index file per indexed attribute. For instance, a CN index file for a database, `example`, holding entries from the suffix `dc=example,dc=com`, is called *ServerRoot*`/slapd-`*ServerID*`/db/example/example_cn.db3`.

What you index depends upon how client applications access directory data. Table 7-1 includes short descriptions of standard index types.

Table 7-1 Standard Index Types

Index Type	**Answers the question...**
Approximate	Which entries have a value that sounds like `foobar` for this attribute?
Browsing	Which entries fit this virtual list view search?
Equality	Which entries have value `foobar` for this attribute?

Table 7-1 Standard Index Types *(Continued)*

Index Type	Answers the question...
International	Which entries match for this international locale?
Presence	Which entries have this attribute?
Substring	Which entries have a value matching `*foo*` for this attribute?

An index file for a particular attribute such as CN may contain multiple types of indexes. For instance, if CN is indexed in the `example` database for equality and for substring matching, then `example_cn.db3` contains both equality and substring indexes.

Refer to the *Sun ONE Directory Server Administration Guide* for:

- An overview of each index type
- Instructions on creating and deleting indexes
- A list of default indexes created by Directory Server
- A list of system indexes required by Directory Server

Default indexes improve search performance in many situations, and include some support for other applications such as messaging. In some cases, you may choose to disable or even delete particular default indexes for performance reasons. System indexes are those on which Directory Server depends. Do not delete or modify them.

Benefits: How Searches Use Indexes

Indexes speed up searches. An index contains a list of values, each associated with a list of entry identifiers corresponding to the value. Directory Server can look up entries quickly using the lists of entry identifiers in indexes. Without an index to manage a list of entries, Directory Server may have to check every entry in a suffix to find matches for a search.

The reason an indexed search may require significantly less processing than an unindexed search becomes evident when search request processing is explained. Here is how Directory Server processes each search request:

1. A client application sends a search request to Directory Server.

2. Directory Server examines the request to ensure the search base corresponds to a suffix it can handle. If not, it returns an error to the client, and may return a referral to another Directory Server instance.

3. Directory Server determines whether it manages an index or indexes appropriate to the search.

   For each such index that exists, Directory Server looks up candidate entries — entries that might be a match for the search request — in the index, as shown in Figure 6-2 on page 113.

   Notice that if no such index exists, Directory Server generates the set of candidate entries from all entries in the database. For large deployments, this step may consume considerable time and system resources, depending on the search.

4. Directory Server examines each candidate entry to determine if it matches the search criteria. Directory Server returns matching entries to the client application as it finds them.

   Directory Server continues examining candidates either until all candidates have been examined, or until it reaches a resource limit such as `nsslapd-lookthroughlimit`, `nsslapd-sizelimit`, or `nsslapd-timelimit`, as described in "Limiting Resources Available to Clients," on page 149.

As is evident from Step 3, indexes can reduce significantly the processing Directory Server must perform to respond to a search request from a client.

Costs: How Updates Affect Indexes

Updates change not only entries themselves, but also indexes referencing the entries. The more references to an entry in indexes, the higher the potential processing cost to modify the indexes during an update. Specifically, Directory Server modifies all impacted indexes as shown in Figure 6-3 on page 115 *before sending acknowledgement of the update to the client application*.

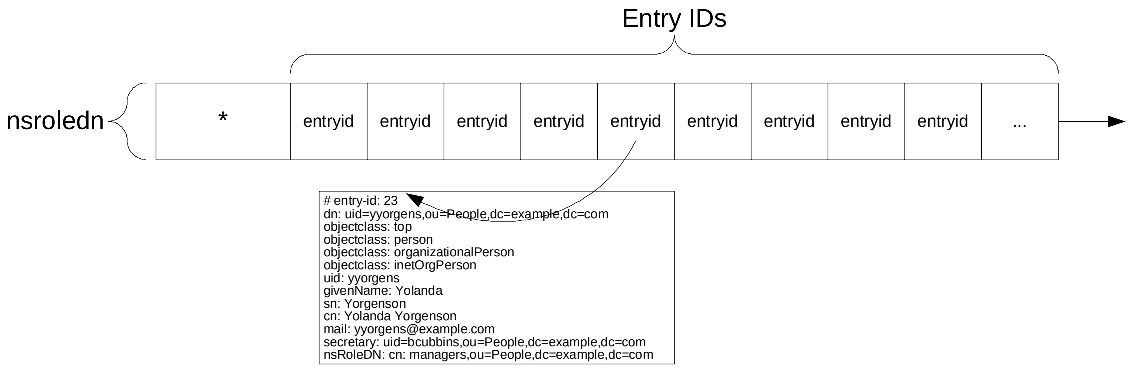
In addition to the processing costs incurred for index maintenance, indexes have a cost in terms of space on disk and potentially space in memory. When optimizing database cache size for searches, as described "Optimizing For Searches," on page 118, you may opt to provide enough memory to hold both entries and indexes in database cache. The larger the indexes, the more space required. 64-bit indexes require somewhat more space than 32-bit indexes, as well.

In general, tuning indexing for an instance of Directory Server means maintaining only those indexes for which the benefits from faster search processing offset the costs of more update processing and of more space needed. Maintaining useful indexes is good practice; maintaining unused indexes for attributes on which clients rarely search is a waste.

Presence Indexes

Figure 7-1 depicts a presence index for the `nsRoleDN` attribute, showing how this index is independent of the attribute value, but simply includes all entries in the database having an `nsRoleDN` attribute. Every value of the attribute matches `*`.

Figure 7-1 Representation of a Presence Index



```
# entry-id: 23
dn: uid=yyorgens,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: yyorgens
givenName: Yolanda
sn: Yorgenson
cn: Yolanda Yorgenson
mail: yyorgens@example.com
secretary: uid=bcubbins,ou=People,dc=example,dc=com
nsRoleDN: cn: managers,ou=People,dc=example,dc=com
```

As shown, the internal `entryid` attribute value allows Directory Server to store a reference to the entry that allows for quick retrieval. Directory Server actually retrieves the entry using the *dbinstance*`_id2entry.db3` index file, where *dbinstance* depends on the database identifier as implied in "About Indexes," on page 125.

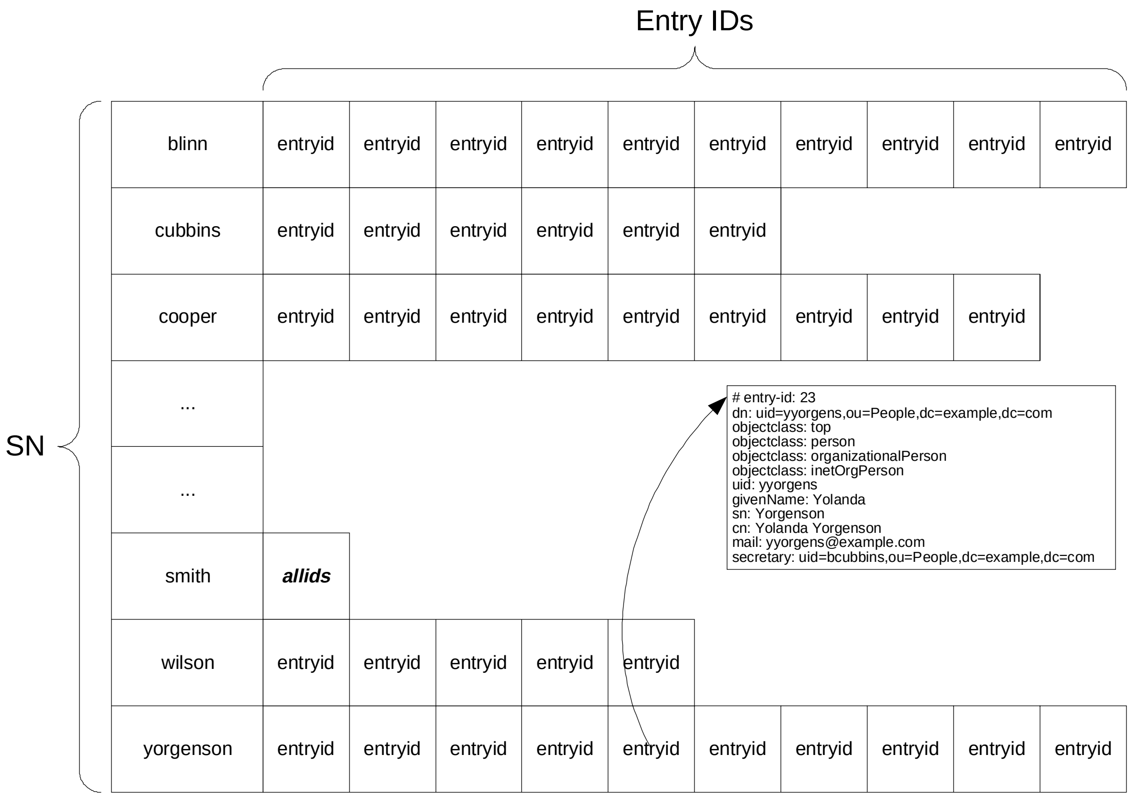
When Directory Server receives an update request for an entry having an attribute indexed for presence, it must determine whether the entry must be removed from the index or not, and must then carry out any necessary modifications before returning acknowledgement of the update to the client application.

The cost of presence indexes is generally lower than for other index types, although the list of entries maintained for a presence index may be long.

Equality Indexes

Figure 7-2 depicts an equality index for the SN (surname) attribute. It shows how this index maintains a list per attribute value of entries having that attribute value for the SN attribute.

Figure 7-2 Representation of an Equality Index



```
# entry-id: 23
dn: uid=yyorgens,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: yyorgens
givenName: Yolanda
sn: Yorgenson
cn: Yolanda Yorgenson
mail: yyorgens@example.com
secretary: uid=bcubbins,ou=People,dc=example,dc=com
```

When Directory Server receives an update request for an entry having an attribute indexed for equality, it must determine whether the entry must be removed from the index or not, determine whether a list must be added or removed from the index, and must then carry out any necessary modifications before returning acknowledgement of the update to the client application.

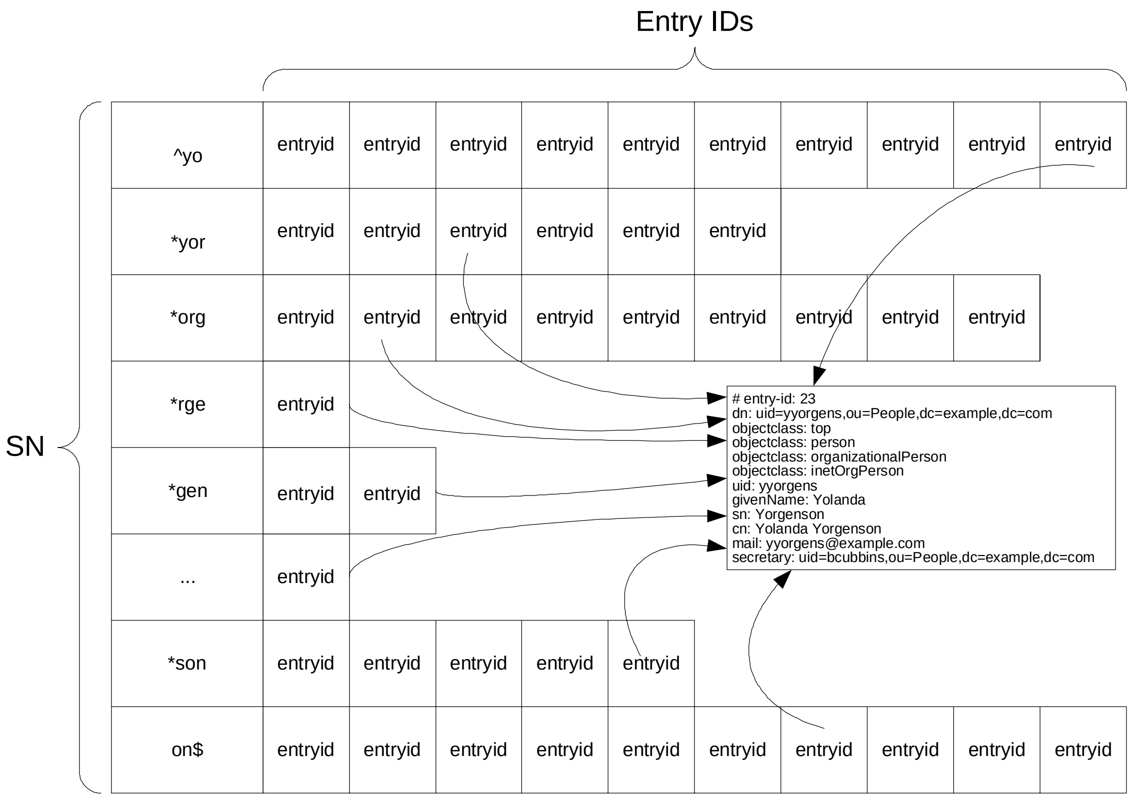
The cost of equality indexes is generally lower than for substring indexes, for example, but higher in terms of space than for presence. Some client applications such as messaging servers may, however, rely on equality indexes for top search performance. Avoid equality indexes for large binary attributes such as photos and encrypted passwords.

Substring Indexes

Figure 7-3 depicts a substring index for the `SN` (surname) attribute. It shows an excerpt of how this index maintains a series of lists per attribute value.

Directory Server indexes substrings such that searches for two-character substrings may be found in the index. A search for `(sn=*ab*)` can therefore be accelerated using an index, for example, but a search for `(sn=*a*)` cannot.

Figure 7-3 Representation of a Substring Index



```
# entry-id: 23
dn: uid=yyorgens,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: yyorgens
givenName: Yolanda
sn: Yorgenson
cn: Yolanda Yorgenson
mail: yyorgens@example.com
secretary: uid=bcubbins,ou=People,dc=example,dc=com
```

Directory Server offers a further optimization allowing initial substring searches of only one character before the wildcard. Thus a search for `(sn=a*)`, but not `(sn=*a*)` or `(sn=*a)`, can also be accelerated when a substring index is available, for example.

Notice that Directory Server builds an index of substrings according to its own built-in rules. These substrings are not configurable by the system administrator.

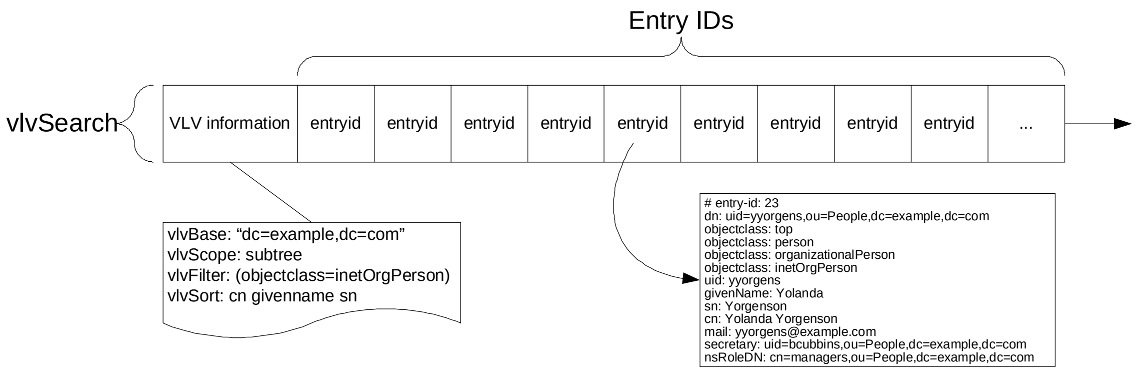
When Directory Server receives an update request for an entry having an attribute indexed for substrings, it must determine whether the entry must be removed from the index, determine whether and how modifications to the entry affect the index, determine whether entry IDs or lists of entry IDs must be added or removed from the index, and must then carry out any necessary modifications before returning acknowledgement of the update to the client application. The number of updates depends on the length of the attribute value string.

Maintaining substring indexes is generally quite costly. As the cost is a function of the length of the string indexed, avoid unnecessary substring indexes, especially for attributes having potentially long string values such as `description`. Substring indexes cannot be applied to binary attributes such as photos.

Browsing (Virtual List View) Indexes

Figure 7-4 depicts a browsing index for a virtual lists view. It shows how this index depends on the virtual list view information. That is, the `vlvBase`, `vlvScope`, `vlvFilter`, and `vlvSort` attribute values for the browsing index. Entry IDs in this type of index are ordered according to the `vlvSort` criteria.

Figure 7-4 Representation of a Browsing Index



```
vlvBase: “dc=example,dc=com”
vlvScope: subtree
vlvFilter: (objectclass=inetOrgPerson)
vlvSort: cn givenname sn
```

```
# entry-id: 23
dn: uid=yyorgens,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: yyorgens
givenName: Yolanda
sn: Yorgenson
cn: Yolanda Yorgenson
mail: yyorgens@example.com
secretary: uid=bcubbins,ou=People,dc=example,dc=com
nsRoleDN: cn=managers,ou=People,dc=example,dc=com
```

When Directory Server receives an update request for an entry matching a `vlvFilter` value, it must determine whether the entry must be removed from the index or not, determine the correct position of the entry in the list, and must then carry out any necessary modifications before returning acknowledgement of the update to the client application.

Approximate Indexes

Directory Server maintains approximate indexes using a variation of the *metaphone* phonetic algorithm. This algorithm breaks down an attribute string value into a rough approximation of its English phonetic pronunciation. Values to match in incoming search requests are handled using the same algorithm. As the algorithm is based loosely on syllables, it is not effective for attributes containing numbers such as telephone numbers.

The algorithm generates a target string for each attribute value string. Costs for this “sounds like” indexing of English-language strings are therefore similar to those for equality indexing.

International Indexes

International indexes use matching rules for particular locales to maintain indexes. Costs for such indexes therefore resemble costs for substring and equality indexes.

Using a custom matching rule server plug-in, you can extend standard support for international and other types of indexing. Refer to the *Sun ONE Directory Server Plug-In API Programming Guide* for more information on custom matching rule plug-ins.

Example: Indexing an Entry

Consider a user entry as shown in being added to a suffix indexed for equality on `uid`, for equality, substring and approximate searches on Common Name (`cn`) and surname (`sn`) attributes, for equality searches on the `mail` attribute, for equality and substring searches on the `telephoneNumber` attribute, and for substring searches on the `description` attribute.

Code Example 7-1 Sample User Entry

```
dn: uid=yyorgens,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: yyorgens
givenName: Yolanda
sn: Yorgenson
cn: Yolanda Yorgenson
mail: yolanda.yorgenson@example.com
telephoneNumber: 1-650-960-1300
description: Business Development Manager, Platinum Partners
```

In adding this entry, Directory Server must modify indexes for `cn`, `sn`, `mail`, `telephoneNumber`, and `description`. Table 7-2 illustrates the expected number of entries.

Table 7-2 Index Updates for Sample User Entry

Attribute	Approximate	Equality	Substring[1]	Total Index Updates
`uid`		1		1
`cn`	1	1	17	19
`sn`	1	1	9	11
`mail`		1		1
`telephoneNumber`		1	11	12
`description`			47	47

1. Substring indexing on strings as long as the `description` string here is not recommended for most deployments.

Notice that the number of substring index updates for the `description` string is larger (47) than the number of updates (44) for all other attributes combined. Also, further modifications to the `description` string may again imply a maximum number of updates or more depending on the new string. In most cases, avoid substring indexing of this volume by not applying substring indexing to long strings such as `description` values.

Tuning Indexing for Performance

In many cases, tuning indexing for performance implies activating indexes to speed up frequent searches, and disactivating indexes that are expensive to maintain and not frequently used.

NOTE Database backups include indexes, and so should match the Directory Server configuration.

After changing how indexes are configured, back up both the configuration and the data.

For large deployments involving replicas dedicated to specific applications, you may opt to configure different indexes for different Directory Server instances. For example, consider a topology with:

- Masters handling writes only
- Hubs handling the replication load to consumers
- Some consumers dedicated to specific applications such as messaging

The masters in this case do not handle searches, so you may choose not to maintain expensive substring indexes on the masters, for example. You may also determine that some other indexes are hardly ever used and can be disactivated.

The hubs essentially receive no client requests other than administrative requests, so you may in this case disactivate all but system indexes required by Directory Server itself.

On specific consumers dedicated to individual applications, you may decide to disactivate all indexes not used by the application. Which indexes you disactivate depends on the searches performed by the particular application.

Allowing Only Indexed Searches

Directory Server makes it possible to prevent costly unindexed searches, returning `LDAP_UNWILLING_TO_PERFORM` to clients requesting an unindexed search.

To prevent unindexed searches against a particular database, set the `nsslapd-require-index` attribute value to on for the database:

```
$ ldapmodify -h host -p port -D "cn=directory manager" -w password
dn: cn=example,cn=ldbm database, cn=plugins, cn=config
changetype: modify
replace: nsslapd-require-index
nsslapd-require-index: on
^D (^Z on Windows systems)
```

The change takes effect immediately. No need to restart Directory Server.

Limiting Index List Length

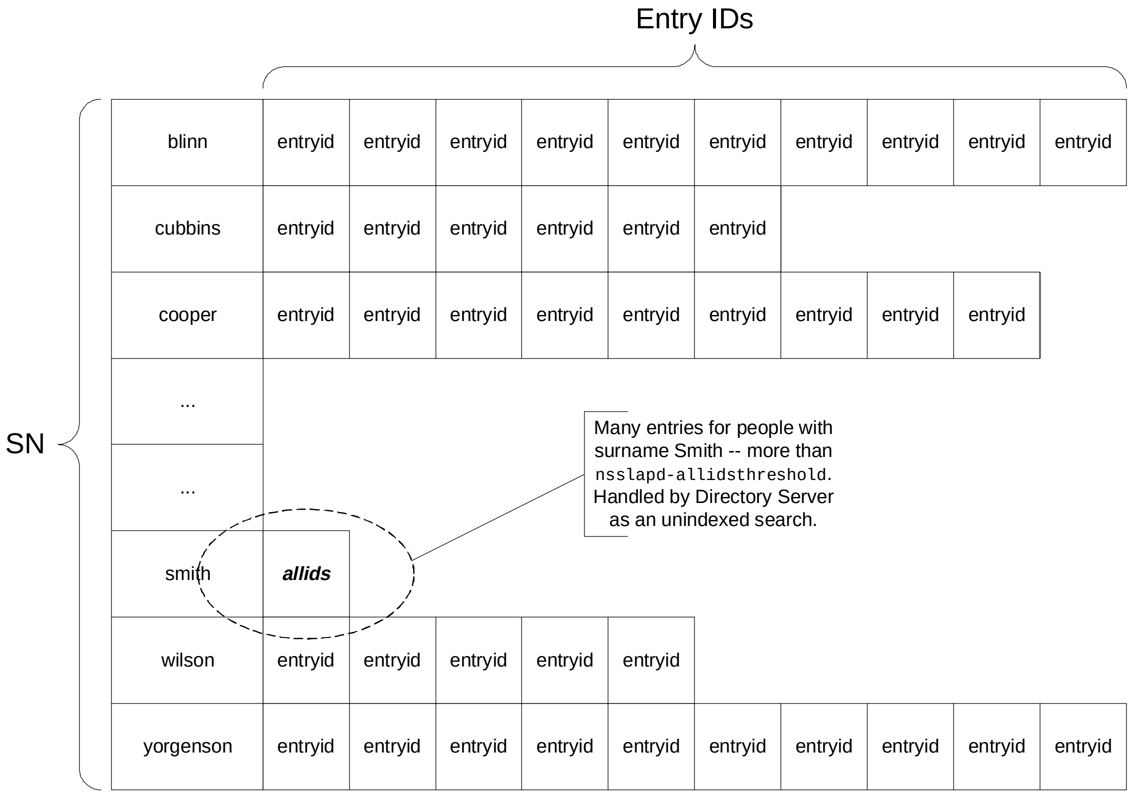
In large and fast growing directory deployments, indexing may reach the point of diminishing returns for a particular index key. At the point of diminishing returns, the list associated with a particular key becomes so long that maintaining the list costs more than performing an occasional unindexed search on that particular key for candidate entries. Imagine for example a very large phone book application equality indexed on surname. Imagine the number of Smiths in the phone book is so large that maintaining an index for Smiths outweighs the lookup benefits. At this point, Directory Server should stop indexing surname for Smith. Directory Server should, however, continue indexing for other surnames.

Directory Server has a mechanism for handling this. You set a configuration attribute to a threshold value. If the number of entries in the list for a particular key gets as large as the value you set, Directory Server replaces the list for the key with a token specifying that an unindexed search should be performed to find candidate entries for that particular key. The value is somewhere near but less than the value for the maximum number of candidate entries checked for a search, set using `nsslapd-lookthroughlimit`, as described in Table 9-1 on page 150.

The mechanism is referred to as the *all IDs threshold*, named after the configuration attribute used to set the global threshold value, `nsslapd-allidsthreshold` on `cn=config,cn=ldbm database,cn=plugins,cn=config`. Notice this value is currently global to the Directory Server instance. It cannot be set differently for different indexes.

Figure 7-5 illustrates the example of indexing on surname with a number of Smiths greater than `nsslapd-allidsthreshold`.

Figure 7-5 Reaching the All IDs Threshold for an Index Key


Entry IDs
SN
blinn
cubbins
cooper
...
...
smith
wilson
yorgenson
entryid
allids
Many entries for people with surname Smith -- more than nsslapd-allidsthreshold. Handled by Directory Server as an unindexed search.

Notice that the threshold affects only one list in the index table. Lists for other keys are not affected.

Symptoms of Inappropriate Index List Size

If clients perform primarily indexed searches and cache sizes are correctly tuned as described in Chapter 6, "Tuning Cache Sizes," yet you still observe poor search performance, an inappropriate threshold value may be the cause. When you observe poor search performance for indexed searches, ensure cache sizes are appropriately tuned first. Next, examine the access log to determine whether Directory Server is reaching the all IDs threshold often.

The `notes=U` flag at the end of an `access` log `RESULT` message indicates Directory Server performed an unindexed search. A previous `SRCH` message for the same connection and operation specifies the search filter used. The following two-line example traces an unindexed search for `(cn=Smith)` returning 10000 entries. Time stamps have been removed from the messages.

```
conn=2 op=1 SRCH base="o=example.com" scope=0 filter="(cn=Smith)"
conn=2 op=1 RESULT err=0 tag=101 nentries=10000 notes=U
```

If you observe many such pairs for searches that should be indexed, you may be able to improve search performance by increasing the threshold.

Changing the Index List Threshold Size

Good values for `nsslapd-allidsthreshold` typically fall in a range around 5 percent of the total number of entries in the directory. For example, the default value of 4000 is generally right for Directory Server instances handling 80,000 entries or less. You may decide to set the value significantly higher than 5 percent of the total if you expect to add large numbers of entries to the directory in the near term, or if you expect the directory to grow considerably. You may also decide to set the threshold differently on consumer replicas supporting many searches than on masters supporting almost only writes. If you plan to reinitialize a large directory from LDIF in the near term, you may even choose to adjust the value for `nsslapd-allidsthreshold` just before reinitialization, as each change to the value of this attribute requires that all indexes be rebuilt. In any case, avoid setting the all IDs threshold very high (above 50,000) even for very large deployments unless you have a good, specific reason for doing so.

Change the all IDs threshold as follows. Note that service is interrupted on the Directory Server instance undergoing the change.

1. Stop the Directory Server instance in question.
2. Export all directory databases to LDIF.

   Refer to the *Sun ONE Directory Server Administration Guide* for details.
3. Carefully adjust the value of the `nsslapd-allidsthreshold` attribute in *ServerRoot*`/slapd-`*ServerID*`/config/dse.ldif`.
4. Reinitialize all directory databases from LDIF.

   Refer to the *Sun ONE Directory Server Administration Guide* for details.

5. If database cache size was tuned for the old all IDs threshold value and the server has adequate physical memory, consider increasing database cache size by 25 percent of the magnitude of the increase to the threshold.

   In other words, if you increase the all IDs threshold from 4000 to 6000, you may choose to increase database cache size by about 12.5 percent to account for increased index list size. Find the optimum size empirically before applying changes to production servers. Refer to Chapter 6, “Tuning Cache Sizes,” for details on database cache tuning.

6. Restart the Directory Server instance.

Troubleshooting Index Fragmentation

Directory Server instances supporting large indexes and high update rates can suffer extreme index key fragmentation. Extreme index key fragmentation may reduce performance even for constant database size. If you have reason to believe extreme index key fragmentation is impacting server performance significantly, consider regenerating the affected indexes to reduce fragmentation.

Refer to the *Sun ONE Directory Server Administration Guide* for details creating indexes.

Chapter 8

Tuning Logging

Directory Server provides several log types, summarized in Table 8-1. This chapter discusses how to handle the different types of logs.

Table 8-1 Types of Logs Used by Directory Server

Log	Type	Use
Access	Flat file	Evaluating directory use patterns, verifying configuration settings, diagnosing access problems.
Audit	Flat file	Providing audit trails for security and data integrity.
Changelog	Database	Enables synchronization between replicas.
Error	Flat file	Debugging directory deployments.
Retro changelog	Database	Permitting backward compatibility with previous versions.
Transaction	Database	Maintaining database integrity.

In high-volume deployments, writing to logs can be disk intensive, resulting in noticeable negative performance impact. Given the potential for I/O bottlenecks inherent with heavy logging in high volume systems, consider placing logs on separate physical disks with separate disk controllers.

Access Logging

The access log contains detailed information about client connections and operations performed. The access log can be indispensable when diagnosing access problems, verifying server configuration settings, and evaluating server usage patterns. The default logging level results, however, in significant disk activity for most deployments, and the volume of disk activity can negatively affect server performance.

Although the access log provides beneficial troubleshooting information, it may become an I/O bottleneck. Consider disabling access logging once the directory is deployed and running without errors or performance problems. When access logging becomes necessary in a production environment, set logging levels to the minimum required level. Additionally, consider placing the access log on its own physical disk or fast disk subsystem having a large I/O buffer. Table 8-2 provides further recommendations for specific attributes.

Table 8-2 Tuning Recommendations for Access Logging

Configuration Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-accesslog`	Specifies the path and filename of the access log file. For low volume deployments, the access log may share a disk with the audit and error logs. For high volume deployments, consider putting the access log on its own disk or disk subsystem, with its own controller. Choose a disk with a large I/O buffer.
`nsslapd-accesslog-level`	Specifies the level of informational logging used. Change to `0`, no access logging, (default `256`, logging for access to an entry) unless a higher level is required.
`nsslapd-accesslog-logbuffering`	Determines whether the access log is buffered. Leave `on` (default) unless you must disable buffering to see access log messages as they are triggered. Disabling buffering can result in a drop in overall performance.

Table 8-2 Tuning Recommendations for Access Logging *(Continued)*

Configuration Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-accesslog-logging-enabled`	Enables and disables access logging. Turn `off` (default is `on`) for maximum performance. If the deployment requires that access logging be enabled, set `nsslapd-accesslog-level` to the lowest acceptable setting, and put the access log on its own disk or disk subsystem. Rotate the access log frequently (each day or week) and use `nsslapd-accesslog-logmaxdiskspace` and `nsslapd-accesslog-logminfreediskspace` to manage disk space use.
`nsslapd-accesslog-logmaxdiskspace`	Specifies maximum disk space that all access logs (current and rotated logs) may consume. Set this value below the total amount of disk space dedicated to access logging. If using the same disk for audit, access, and error logging, ensure sufficient disk space for all three. If the access log resides on its own disk, set this variable to the size of the disk.
`nsslapd-accesslog-logminfreediskspace`	Specifies minimum free disk space allowed before old logs are purged. When the amount of free disk space falls below the value specified on this attribute, the oldest access logs are deleted until enough disk space is freed to correspond to the setting for this attribute. If the access logs cannot be written because the disk is full, the server shuts down.

Refer to the *Sun ONE Directory Server Reference Manual* for details concerning individual configuration attributes.

Sun ONE Directory Server Resource Kit documentation covers extracting information from the access log. Refer to "Downloading Directory Server Tools," on page 12, for more information.

Audit Logging

The audit log contains detailed information about all changes made to each database as well as to server configuration. Audit logging is disabled by default.

When enabled in deployments having high modify volume, enabling audit logging causes a very noticeable overall drop in performance. Unless the deployment requires it, leave audit logging disabled. For large or high volume deployments that require audit logging, consider allocating a separate disk on a separate controller to the audit log. Table 8-3 provides further recommendations for specific attributes.

Table 8-3 Tuning Recommendations for Audit Logging

Configuration Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-auditlog`	Specifies the path and filename of the audit log file. For low volume deployments, the audit log may share a disk with the access and error logs. For high volume deployments, consider putting the audit log on its own disk, with its own controller. Choose a disk with a large I/O buffer.
`nsslapd-auditlog-logging-enabled`	Enables and disables audit logging. Leave `off` (default setting) unless audit logging is required.
`nsslapd-auditlog-logmaxdiskspace`	Specifies maximum disk space that all audit logs (current and rotated logs) may consume. Set this value below the total amount of disk space dedicated to audit logging. If using the same disk for audit, access, and error logging, ensure sufficient disk space for all three. If the audit log resides on its own disk, set this variable to the size of the disk.

Table 8-3 Tuning Recommendations for Audit Logging *(Continued)*

Configuration Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-auditlog-logminfreediskspace`	Specifies minimum free disk space allowed before old logs are purged. When the amount of free disk space falls below the value specified on this attribute, the oldest audit logs are deleted until enough disk space is freed to correspond to the setting for this attribute. If the audit logs cannot be written because the disk is full, the server shuts down.

Refer to the *Sun ONE Directory Server Reference Manual* for details concerning individual configuration attributes.

Error Logging

The error log for a Directory Server instance contains detailed error, warning, and informational messages encountered during normal server operation. The low default logging level produces relatively little disk activity.

When log level is set higher to generate debugging information, however, Directory Server may begin writing large numbers of messages to disk. The write load can result in a very noticeable overall drop in performance. To avoid a drop in performance, increase log levels progressively, component by component, instead of activating log levels for all components at once.

The error log does not support log buffering. All messages are flushed to disk immediately. For large or high volume deployments, consider allocating a separate disk on a separate controller for the error log, used whenever debugging becomes necessary. Table 8-4 provides further recommendations for specific attributes.

Table 8-4 Tuning Recommendations for Error Logging

Configuration Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-errorlog`	Specifies the path and filename of the error log file. For low volume deployments, the error log may share a disk with the access and audit logs. For high volume deployments, consider putting the error log on its own disk, with its own controller. Choose a disk with a large I/O buffer.
`nsslapd-errorlog-logging-enabled`	Enables and disables error logging. Leave `on` (default setting).
`nsslapd-errorlog-logmaxdiskspace`	Specifies maximum disk space that all error logs (current and rotated logs) may consume. Set this value below the total amount of disk space dedicated to error logging. If using the same disk for audit, access, and error logging, ensure sufficient disk space for all three. If the error log resides on its own disk, set this variable to the size of the disk.
`nsslapd-errorlog-logminfreediskspace`	Specifies minimum free disk space allowed before old logs are purged. When the amount of free disk space falls below the value specified on this attribute, the oldest error logs are deleted until enough disk space is freed to correspond to the setting for this attribute. If the error logs cannot be written because the disk is full, the server shuts down.
`nsslapd-infolog-area`	Specifies the components for which informational messages are logged. Leave at `0` (default) unless debugging a component. Avoid setting for more than one component at a time on production servers.

Table 8-4 Tuning Recommendations for Error Logging *(Continued)*

Configuration Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-infolog-level`	Specifies the level of informational logging used. Leave at `0` (default) unless debugging a component for which setting `nsslapd-infolog-area` alone fails to generate sufficient detail.

Refer to the *Sun ONE Directory Server Reference Manual* for details concerning individual configuration attributes.

Multi-Master Replication Change Logging

Directory Server uses a replication changelog to enable synchronization between replicas. Refer to the *Sun ONE Directory Server Deployment Guide* for an discussion of the changelog and to the *Sun ONE Directory Server Reference Manual* for configuration details. Table 8-5 provides further recommendations for specific attributes.

Table 8-5 Tuning Recommendations for Multi-Master Change Logging

Configuration Entry DN and Configuration Attribute	Short Description and Tuning Recommendations

`dn: cn=changelog5,cn=config`
 	
`nsslapd-cachememsize`
	Specifies the changelog database cache size.
	
	Consider changing this from the default of 10 MB for high volume deployments.

`dn: cn=changelog5,cn=config`
 	
`nsslapd-changelogdir`
	Specifies the path and filename of the changelog database.
	
	Consider putting the changelog on its own disk or disk subsystem, with its own controller. A large I/O buffer can help.

`dn: cn=changelog5,cn=config`
 	
`nsslapd-changemaxage`
	Specifies the maximum age for entries in the changelog.
	
	Change this from `0` (default, indicating no maximum) to an interval after which replicated servers are fully synchronized and the changelog may be trimmed.

Table 8-5 Tuning Recommendations for Multi-Master Change Logging *(Continued)*

Configuration Entry DN and Configuration Attribute	Short Description and Tuning Recommendations

`dn: cn=changelog5,cn=config`
`nsslapd-changemaxentries`
	Specifies the maximum number of entries in the changelog.
	Change this from `0` (default, indicating no maximum) to a number sufficient to allow replicated servers to become fully synchronized before the changelog is trimmed.

`dn: cn=changelog5,cn=config`
`nsslapd-cachesize`
	Specifies the maximum number of entries in the changelog database cache.
	Change this from `-1` (default, indicating no maximum) to a maximum number of entries retained in the changelog before entries are flushed.

Refer to the *Sun ONE Directory Server Reference Manual* for details concerning individual configuration attributes.

Retro Change Logging

Directory Server ships with a retro changelog plug-in that you may enable to record changes on a supplier server in a format compatible with Directory Server 4.x releases and accessible through LDAP. The retro changelog plug-in is disabled by default and should not be enabled unless required for compatibility reasons. Refer to the *Sun ONE Directory Server Reference Manual* for details. Table 8-6 provides further recommendations for specific attributes.

Table 8-6 Tuning Recommendations for Retro Change Logging

Configuration Entry DN and Configuration Attribute	Short Description and Tuning Recommendations

`dn: cn=Retro Changelog Plugin,cn=plugins,cn=config`
`nsslapd-changelogdir`
	Specifies the path and filename of the retro changelog.
	Consider putting the retro changelog on its own disk or disk subsystem, with its own controller. A large I/O buffer can help.

Table 8-6 Tuning Recommendations for Retro Change Logging *(Continued)*

Configuration Entry DN and Configuration Attribute	Short Description and Tuning Recommendations
`dn: cn=Retro Changelog Plugin,cn=plugins,cn=config`	Specifies the maximum age for entries in the retro changelog.
`nsslapd-changelogmaxage`	Change this from `0` (default, indicating no maximum) to an interval after which clients using the retro changelog have processed the log entries generated.
`dn: cn=Retro Changelog Plugin,cn=plugins,cn=config`	Specifies the maximum number of entries in the retro changelog.
`nsslapd-changelogmaxentries`	Change this from `0` (default, indicating no maximum) to a maximum number of entries retained in the retro changelog before trimming.

Refer to the *Sun ONE Directory Server Reference Manual* for details concerning individual configuration attributes.

Transaction Logging

Directory Server maintains database integrity through transaction logging. Upon accepting an update operation — `add`, `modify`, `delete`, or `modrdn` — Directory Server writes a log message about the operation to the transaction log. Durable transaction logging, enabled by default, ensures data integrity. It does so by ensuring each update operation is committed to the transaction log on disk before the result code for the update operation is returned to the client application. In the event of a system crash, Directory Server uses the transaction log to recover the database. As the transaction log aids in the recovery of a database shut down abnormally, consider storing the transaction log and directory database on separate disk subsystems.

Transaction logging is extremely disk intensive, especially with durability turned on. It is likely to be the major bottleneck for update performance. In addition to protecting data integrity better in the event of a system crash, storing the transaction log and database on separate RAID systems such as Sun StorEdge disk arrays can boost update performance. Table 8-7 provides further recommendations for specific attributes.

Table 8-7 Tuning Recommendations for Transaction Logging

Configuration Entry DN and Configuration Attribute	Short Description and Tuning Recommendations

`dn: cn=config,cn=ldbm database,cn=plugins,cn=config`
 	
`nsslapd-db-checkpoint-interval`
	Specifies how often Directory Server checkpoints the transaction log, ensures the entire database system is synchronized to disk, and cleans up transaction logs.
	
	Leave at `60` (default interval in seconds) unless database performance optimization based on empirical testing calls for a different value. Increasing the value of this attribute may result in a performance boost for update operations, but also means that recovery after disorderly shutdown takes longer, and that the transaction log uses more disk space.

`dn: cn=config,cn=ldbm database,cn=plugins,cn=config`
 	
`nsslapd-db-durable-transaction`
	Specifies whether update operations are committed to the transaction log on disk before result codes are sent to clients.
	
	Leave `on` (default) for deployments requiring a high level of data integrity. Durable transaction logging may be disabled for some deployments to boost performance. When it is disabled, however, log messages flushed to the file system but not yet to disk may be lost in the event of a system crash. This means that with durable transaction logging `off`, some updates may be unrecoverable even after the client receives a successful update result code.

`dn: cn=config,cn=ldbm database,cn=plugins,cn=config`
 	
`nsslapd-db-logdirectory`
	Specifies the path and filename of the transaction log.
	
	Consider storing the transaction log on its own very fast disk or disk subsystem, with its own controller.

Refer to the *Sun ONE Directory Server Reference Manual* for details concerning individual configuration attributes.

Chapter 9

Managing Use of Other Resources

After optimizing cache size, attribute value indexing, and log management, it may prove useful to tune how Directory Server limits resources made available to client applications, and how Directory Server makes use of system resources. It may also prove useful to reconfigure and even disable some features offered as Directory Server plug-ins.

Limiting Resources Available to Clients

Default configuration may allow client applications to use more Directory Server resources than are actually required. This may leave the door open to accidentally or intentionally abusive client applications negatively impacting server performance, by opening many connections then leaving them idle or unused, launching costly and unnecessary unindexed searches, or storing enormous and unplanned for binary attribute values in the directory.

In some deployment situations, it is not advisable to modify the default configuration. For deployments in which you opt not to change the configuration attribute values mentioned in this section, consider using Sun ONE Directory Proxy Server software to set limits externally, and to help protect against denial of service attacks.

In some deployment situations, one instance of Directory Server must support both directory-intensive client applications such as messaging servers and occasional directory clients such as user mail applications. In such situations, consider using bind DN-based resource limits as described in the *Sun ONE Directory Server Administration Guide* to raise individual limits for directory-intensive applications.

The recommendations in Table 9-1 address settings for limiting resources available to all client applications. These limits do not apply to the Directory Manager user, so ensure client applications do not connect as the Directory Manager user.

Table 9-1 Tuning Recommendations for Limiting Resources Available to Clients

Configuration Entry DN and Attribute	Short Description and Tuning Recommendations

`dn: cn=config`
 	
`nsslapd-idletimeout`
	Sets the time in seconds after which Directory Server closes an idle client connection. Here *idle* means that the connection remains open, yet no operations are requested. By default, no time limit is set.
	
	Some applications, such as messaging servers, may open a pool of connections that remain idle when traffic is low, but that should not be closed. Ideally, you might dedicate a replica to support the application in this case. If that is not possible, consider bind DN-based limits.
	
	In any case, set this value high enough not to close connections that other applications expect to remain open, but set it low enough that connections cannot be left idle abusively. Consider using 120 seconds (2 minutes) as a starting point for optimization tests.

`dn: cn=config`
 	
`nsslapd-ioblocktimeout`
	Sets the time in milliseconds after which Directory Server closes a stalled client connection. Here *stalled* means that the the server is blocked either sending output to the client or reading input from the client.
	
	For Directory Server instances particularly exposed to denial of service attacks, consider lowering this value from the default of 1,800,000 milliseconds (30 minutes).

`dn: cn=config,cn=ldbm database,cn=plugins,cn=config`
 	
`nsslapd-lookthroughlimit`
	Sets the maximum number of candidate entries checked for matches during a search.
	
	Some applications, such as messaging servers, may need to search the entire directory. Ideally, you might dedicate a replica to support the application in this case. If that is not possible, consider bind DN-based limits.
	
	In any case, consider lowering this value from the default of 5000 entries, but not below the threshold value of `nsslapd-sizelimit`.

Table 9-1 Tuning Recommendations for Limiting Resources Available to Clients *(Continued)*

Configuration Entry DN and Attribute	Short Description and Tuning Recommendations

`dn: cn=config`
 	
`nsslapd-maxbersize`
	Sets the maximum size in bytes for an incoming message. Directory Server rejects requests to add entries larger than this limit.
	
	If you are confident you can accurately anticipate maximum entry size for your directory data, consider changing this value from the default of 2097152 (2 MB) to the size of the largest expected directory entry.

`dn: cn=config`
 	
`nsslapd-maxthreadsperconn`
	Sets the maximum number of threads per client connection.
	
	Some applications, such as messaging servers, may open a pool of connections and may issue many requests on each connection. Ideally, you might dedicate a replica to support the application in this case. If that is not possible, consider bind DN-based limits.
	
	If you anticipate that some applications may perform many requests per connection, consider increasing this value from the default of 5, but do not increase it to more than 10. It is typically not advisable to specify more than 10 threads per connection.

`dn: cn=config`
 	
`nsslapd-sizelimit`
	Sets the maximum number of entries Directory Server returns in response to a search request.
	
	Some applications, such as messaging servers, may need to search the entire directory. Ideally, you might dedicate a replica to support the application in this case. If that is not possible, consider bind DN-based limits.
	
	In any case, consider lowering this value from the default of 2000 entries.

Table 9-1 Tuning Recommendations for Limiting Resources Available to Clients *(Continued)*

Configuration Entry DN and Attribute	Short Description and Tuning Recommendations

`dn: cn=config`
 	
`nsslapd-timelimit`
	Sets the maximum number of seconds Directory Server allows for handling a search request.
	
	Some applications, such as messaging servers, may need to perform very large searches. Ideally, you might dedicate a replica to support the application in this case. If that is not possible, consider bind DN-based limits.
	
	In any case, set this value as low as you can and still meet deployment requirements. The default value of 3600 seconds (1 hour) is larger than necessary for many deployments. Consider using 600 seconds (10 minutes) as a starting point for optimization tests.

Refer to the *Sun ONE Directory Server Reference Manual* for details concerning individual configuration attributes.

Using Available System Resources

Depending on deployment requirements, you may choose to tune how a Directory Server instance uses system and network resources, how access control is managed, and how server plug-ins are configured. The recommendations in Table 9-2 address settings for system resources.

Table 9-2 Tuning Recommendations for Configuring Use of System Resources

Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-listenhost`	Sets the hostname for the IP interface on which Directory Server listens. This attribute is single-valued. Default behavior is to listen on all interfaces. The default behavior is adapted for high volume deployments using redundant network interfaces for availability and throughput. Consider setting this value when deploying on a multihomed system, or when listening only for IPv4 or IPv6 traffic on a system supporting each protocol through a separate interface. Consider setting `nsslapd-securelistenhost` when using SSL.

Table 9-2 Tuning Recommendations for Configuring Use of System Resources *(Continued)*

Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-maxdescriptors`	Sets the maximum number of file descriptors Directory Server attempts to use. Directory Server uses file descriptors to handle client connections, and to maintain files internally. If the error log indicates Directory Server sometimes stops listening for new connections because not enough file descriptors are available, increasing the value of this attribute may increase the number of client connections Directory Server can handle simultaneously. If you have increased the number of file descriptors available on the system as described in "File Descriptors," on page 103, then set the value of this attribute accordingly. The value of this attribute should be less than or equal to the maximum number of file descriptors available on the system.
`nsslapd-nagle`	Sets whether to delay sending of TCP packets at the socket-level. Leave `off` (default) to prevent protocol-level delays in sending results to client applications.

Table 9-2 Tuning Recommendations for Configuring Use of System Resources *(Continued)*

Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-reservedescriptors`	Sets the number of file descriptors Directory Server maintains to manage indexing, replication and other internal processing. Directory Server does not use such file descriptors to handle client connections. Consider increasing the value of this attribute from the default of 64 if all of the following are true. • Directory Server replicates to more than 10 consumers or Directory Server maintains more than 30 index files. • Directory Server handles a large number of client connections. • Messages in the error log suggest Directory Server is running out of file descriptors for operations *not* related to client connections. Notice that as the number of reserved file descriptors increases, the number of file descriptors available to handle client connections decreases. If you increase the value of this attribute, consider increasing the number of file descriptors available on the system, and increasing the value of `nsslapd-maxdescriptors`. If you decide to change this attribute, for a first estimate of the number of file descriptors to reserve, try setting the value of `nsslapd-reservedescriptors` to: `20 + 4 * (number of databases) + (total number of indexes) + (value of nsoperationconnectionslimit) * (number of chaining backends) + ReplDescriptors + PTADescriptors + SSLDescriptors` Where *ReplDescriptors* = number of supplier replica + 8 if replication is used, *PTADescriptors* is 3 if the Pass Through Authentication (PTA) plug-in is enabled (0 otherwise), and *SSLDescriptors* is 5 if SSL is used (0 otherwise). The number of databases is the same as the number of suffixes for the instance, unless the instance is configured to use more than one database per suffix. Verify estimates through empirical testing.
`nsslapd-securelistenhost`	Sets the hostname for the IP interface on which Directory Server listens for SSL connections. This attribute is single-valued. Default behavior is to listen on all interfaces. Consider this attribute in the same way as `nsslapd-listenhost`.

Table 9-2 Tuning Recommendations for Configuring Use of System Resources *(Continued)*

Attribute (on dn: cn=config)	Short Description and Tuning Recommendations
`nsslapd-threadnumber`	Sets the number of threads Directory Server uses. Consider adjusting the value of this attribute if any of the following are true: • Client applications perform many time-consuming operations such as updates or complex searches simultaneously. • Directory Server supports many simultaneous client connections. • Directory Server handles more than 5,000,000 entries. Multiprocessor systems can sustain larger thread pools than single processor systems. As a first estimate when optimizing the value of this attribute, use two times the number of processors or 20 + number of simultaneous updates. Consider also adjusting the maximum number of threads per client connection, `nsslapd-maxthreadsperconn`, as discussed in Table 9-1. The maximum number of these threads handling client connections cannot exceed the maximum number of file descriptors available on the system. In some cases, it may prove useful to *reduce*, rather than increase, the value of this attribute. Verify estimates through empirical testing. Results depend not only on the particular deployment situation but also on the underlying system.

Refer to the *Sun ONE Directory Server Reference Manual* for details concerning individual configuration attributes.

Managing Access Control

Directory Server now offers performance and scalability improvements for Access Control Instructions (ACIs) such as better memory management and support for macro ACIs. Improvements notwithstanding, Directory Server uses significant system resources to evaluate complex ACIs. Extensive use of complex ACIs can therefore negatively impact performance.

Macro ACIs help you limit the number of ACIs used. By limiting the number of ACIs, you render access control easier to manage and reduce the load on the system. Macros are placeholders that represent a DN, or a portion of a DN, in an ACI. A macro can be used in an ACI target, in an ACI bind rule, or in both. When Directory Server receives a request, it checks which ACI macros match against the resource targeted for the resulting operation. If a macro matches, Directory Server replaces it with the value of the actual DN. Directory Server then evaluates the ACI normally. For more information on ACIs, refer to the *Sun ONE Directory Server Administration Guide*.

Testing has demonstrated that Directory Server can support more than 50,000 ACIs. The impact on performance for various deployment scenarios is currently under analysis. Keep the number of ACIs as small as possible to limit negative impact on performance, and to reduce the complexity of managing access controls. For deployments involving complex ACI environments, consider using Sun ONE Directory Proxy Server to provide some access control features.

Configuring Server Plug-Ins

Directory Server implements many key features such as access control, replication, syntax checking, and attribute uniqueness using plug-ins. In the context of a particular deployment, you may find it useful to reconfigure some plug-ins. The recommendations in Table 9-3 address settings for some standard plug-ins.

Table 9-3 Tuning Recommendations for Some Standard Plug-Ins

Name and DN	Short Description and Tuning Recommendations
7-Bit Check Plug-In `dn: cn=7-bit check,cn=plugins,cn=config`	Allows Directory Server to check that attribute values are 7-bit clean. That is, that attribute values provided contain only those characters that fit in 7-bit encoding. You may choose to disable this plug-in (default `on`) if the infrastructure is designed to support wider encodings such as Japanese characters, for example.
Legacy Replication Plug-In `dn: cn=Legacy Replication Plugin,cn=plugins,cn=config`	Allows Directory Server to function as a consumer of a 4.x supplier. Unless you intend to use Directory Server as a consumer of a 4.x supplier during an upgrade for example, turn this plug-in off (`on` by default in case 4.x replication capabilities are required).

Table 9-3 Tuning Recommendations for Some Standard Plug-Ins *(Continued)*

Name and DN	Short Description and Tuning Recommendations
Referential Integrity Plug-In `dn: cn=referential integrity postoperation,cn=plugins,cn=config`	Allows Directory Server to ensure relationships between related entries are maintained. For example, when a user entry is removed from the directory or renamed, the groups to which the user belonged are updated as needed without manual intervention. Enable and configure this plug-in on all masters. If you opt to enable the plug-in, create equality indexes for all attributes configured for use with the plug-in. The plug-in uses such indexes when searching for entries to update. Without equality indexes for the attributes it uses, the plug-in must perform costly unindexed searches that have negative impact on performance. Refer to the *Sun ONE Directory Server Administration Guide* for instructions on configuring and enabling the plug-in.

Refer to the *Sun ONE Directory Server Reference Manual* for details concerning individual configuration attributes.

Appendix A

Installed Product Layout

This appendix summarizes product software layout after a typical installation. Of the files installed, only those listed here and discussed in the product documentation belong to the supported public product interface.

NOTE Examples shown here reflect a product installation for the Solaris Operating Environment. File names and extensions may differ for installations on other platforms.

After installing the Solaris packaged version of the product, you can also obtain a full list of installed path names for a particular package using the `pkgchk`(1M) utility, `pkgchk -v` *package-name*.

Some platforms such as the Solaris Operating Environment provide integrated tools for managing directory services. Sun ONE Directory Server also provides tools. Refer to the *Sun ONE Directory Server Administration Guide* and *Sun ONE Directory Server Reference Manual* for details on the tools listed here.

The *ServerRoot* Directory

The *ServerRoot* directory contains several system administration utilities. To determine the path corresponding to the *ServerRoot* directory for your platform, configuration, and installation, refer to "Default Paths and Filenames," on page 10.

Table A-1 Utilities Under *ServerRoot*

Utility	Remarks
ServerRoot/`restart-admin`	Restart administration server
ServerRoot/`start-admin`	Start administration server

Table A-1 Utilities Under *ServerRoot (Continued)*

Utility	Remarks
ServerRoot/`startconsole`	Start Sun ONE Server Console
ServerRoot/`stop-admin`	Stop administration server
ServerRoot/`uninstall`	Uninstall product software

The *ServerRoot*/`bin` directory contains product binaries and configuration templates used internally when creating a server instance.

Table A-2 Files Under *ServerRoot*/`bin`

File	Remarks
ServerRoot/`bin/`	Internal use, except for the following:
ServerRoot/`bin/admin/admconfig`	Configure administration server
ServerRoot/`bin/https/bin/ns-httpd`	Sun ONE Administration Server
ServerRoot/`bin/https/bin/uxwdog`	Administration server watchdog
ServerRoot/`bin/slapd/server/ns-ldapagt`	LDAP-based SNMP subagent
ServerRoot/`bin/slapd/server/ns-slapd`	Sun ONE Directory Server

The *ServerRoot*/`lib` directory contains product libraries, including plug-ins.

Table A-3 Libraries Under *ServerRoot*/`lib`

Library	Remarks
ServerRoot/`lib/`	Internal use and plug-ins
ServerRoot/`lib/libnspr4.so`	NSPR, version 4.x

The *ServerRoot*/`manual` directory contains support for console online help.

Table A-4 Support for Online Help Under *ServerRoot*/`manual`

Directory	Remarks
ServerRoot/`manual/`	Support for online help

The *ServerRoot*/`plugins` directory contains sample server plug-ins, header files for plug-in development, and plug-ins for SNMP support.

Table A-5 Support for Plug-Ins Under *ServerRoot*/`plugins`

Directory or File	Remarks
ServerRoot/`plugins/`	Samples, headers, SNMP support
ServerRoot/`plugins/slapd/slapi/examples/`	Sample plug-ins
ServerRoot/`plugins/slapd/slapi/include/`	Plug-in header files
ServerRoot/`plugins/snmp/magt/magt`	Configure management agent
ServerRoot/`plugins/snmp/mibs/`	SNMP MIBs
ServerRoot/`plugins/snmp/sagt/sagt`	Configure SNMP agent

The *ServerRoot*/`shared/bin` directory contains tools for managing the server.

Table A-6 Tools and Clients Under *ServerRoot*/`shared/bin`

Directory or File	Remarks
ServerRoot/`shared/bin`	Internal use, except for the following
ServerRoot/`shared/bin/admin_ip.pl`	Change IP address
ServerRoot/`shared/bin/entrycmp`	Compare entries for replication
ServerRoot/`shared/bin/fildif`	Dump filtered LDIF
ServerRoot/`shared/bin/insync`	Check replication synchronization
ServerRoot/`shared/bin/ldapcompare`	Compare attribute value
ServerRoot/`shared/bin/ldapdelete`	Delete directory entry
ServerRoot/`shared/bin/ldapmodify`	Modify directory entry
ServerRoot/`shared/bin/ldapsearch`	Find directory entries
ServerRoot/`shared/bin/modutil`	Manage PKCS #11 modules
ServerRoot/`shared/bin/uconv`	Convert from ISO to UTF-8
ServerRoot/`shared/bin/repldisc`	Discover replication topology

The *ServerRoot*/`shared/config` directory contains a configuration file for mapping certificates to directory entries.

Table A-7 Certificate Mapping Configuration File Under *ServerRoot*/`shared/config`

Directory or File	Remarks
ServerRoot/`shared/config`	Internal use, except for the following
ServerRoot/`shared/config/certmap.conf`	Map certificates to entries

The *ServerRoot*/`setup5` directory contains sample templates for silent installation and uninstallation.

Table A-8 Silent Installation and Uninstallation Templates Under *ServerRoot*/`setup5`

Directory or File	Remarks
ServerRoot/`setup5`	Internal use, except for the following
ServerRoot/`setup5/typical.ins`	Silent installation template file
ServerRoot/`setup5/uninstall.ins`	Silent uninstallation template file

The Server Instance Directory

The `slapd-`*ServerID* directory contains files corresponding to the server instance *ServerID*. The *ServerRoot*/`slapd-`*ServerID* directory itself contains several scripts for command-line administration.

Table A-9 Server Instance Scripts

Scripts	Remarks
ServerRoot/`slapd-`*ServerID*/	Server instance
ServerRoot/`slapd-`*ServerID*/`bak2db`	Restore database (offline)
ServerRoot/`slapd-`*ServerID*/`bak2db.pl`	Restore database (online)
ServerRoot/`slapd-`*ServerID*/`db2bak`	Backup database (offline)
ServerRoot/`slapd-`*ServerID*/`db2bak.pl`	Backup database (online)
ServerRoot/`slapd-`*ServerID*/`db2index.pl`	Generate index (online)
ServerRoot/`slapd-`*ServerID*/`db2ldif`	Dump database to LDIF (offline)

Table A-9 Server Instance Scripts *(Continued)*

Scripts	**Remarks**
ServerRoot/`slapd-`*ServerID*/`db2ldif.pl`	Dump database to LDIF (online)
ServerRoot/`slapd-`*ServerID*/`getpwenc`	Print encrypted password
ServerRoot/`slapd-`*ServerID*/`ldif2db`	Import LDIF (offline)
ServerRoot/`slapd-`*ServerID*/`ldif2db.pl`	Import LDIF (online)
ServerRoot/`slapd-`*ServerID*/`ldif2ldap`	Import LDIF over LDAP
ServerRoot/`slapd-`*ServerID*/`monitor`	Retrieve monitoring information
ServerRoot/`slapd-`*ServerID*/`ns-accountstatus.pl`	Establish account status
ServerRoot/`slapd-`*ServerID*/`ns-activate.pl`	Activate entries
ServerRoot/`slapd-`*ServerID*/`ns-inactivate.pl`	Inactivate entries
ServerRoot/`slapd-`*ServerID*/`restart-slapd`	Restart directory server
ServerRoot/`slapd-`*ServerID*/`restoreconfig`	Restore administration server configuration
ServerRoot/`slapd-`*ServerID*/`saveconfig`	Save administration server configuration
ServerRoot/`slapd-`*ServerID*/`start-slapd`	Start directory server
ServerRoot/`slapd-`*ServerID*/`stop-slapd`	Stop directory server
ServerRoot/`slapd-`*ServerID*/`suffix2instance`	Map suffix to backend
ServerRoot/`slapd-`*ServerID*/`vlvindex`	Create virtual list view index

Subdirectories of *ServerRoot*/`slapd-`*ServerID* contain configuration, log, and backup data.

Table A-10 Server Instance Subdirectories

Directory	**Remarks**
ServerRoot/`slapd-`*ServerID*/	Server instance
ServerRoot/`slapd-`*ServerID*/`bak/`	Directory database backup
ServerRoot/`slapd-`*ServerID*/`confbak/`	Administration server configuration backup

Table A-10 Server Instance Subdirectories *(Continued)*

Directory	Remarks
ServerRoot/`slapd-`*ServerID*/`conf_bk/`	Directory server configuration backup
ServerRoot/`slapd-`*ServerID*/`config/`	Directory server configuration
ServerRoot/`slapd-`*ServerID*/`config/schema/`	Directory schema configuration
ServerRoot/`slapd-`*ServerID*/`db/`	Directory databases
ServerRoot/`slapd-`*ServerID*/`ldif/`	Sample LDIF files
ServerRoot/`slapd-`*ServerID*/`locks/`	Run time process locks
ServerRoot/`slapd-`*ServerID*/`logs/`	Server log files
ServerRoot/`slapd-`*ServerID*/`tmp/`	Run time temporary files

Manage your server instance using the tools provided. Do not modify directory contents manually.

Internal Use Only

The content of the following are used internally by the Directory Server. These internal components do not belong to the supported public interface.

- *ServerRoot*/`adminacl/`
- *ServerRoot*/`admin-serv/`
- *ServerRoot*/`admserv`
- *ServerRoot*/`alias/`
- *ServerRoot*/`dist/`
- *ServerRoot*/`httpacl/`
- *ServerRoot*/`include/`
- *ServerRoot*/`install/`
- *ServerRoot*/`java/`
- *ServerRoot*/`userdb/`

Do not modify these directories or their contents.

Appendix B

Using the Sun Crypto Accelerator Board

This appendix provides instructions on using a Sun Crypto Accelerator board with Directory Server to enhance performance for connections using the Secure Sockets Layer (SSL) protocol with certificate-based authentication.

Before You Start

Table B-1 covers items that must be completed before attempting to use the Sun Crypto Accelerator board to enhance SSL connection performance.

Table B-1 Prerequisites to Using the Board

Prerequisite	**Remarks**
Board installation	Refer to the product documentation provided for the board when installing the hardware, drivers, patches, and administrative utilities on the host.
Directory Server installation	Refer to Chapter 1, "Installing Sun ONE Directory Server," for instructions.
Server cert. (PKCS#12 format)	Obtain a server certificate for Directory Server as a `.p12` file
CA cert. (PEM format)	Obtain the CA certificate for your Certificate Authority (CA) as a Privacy Enhanced Mail (PEM) format file.

Refer to *Sun ONE Server Console Server Management Guide* both for a discussion of the SSL protocol itself and of SSL certificates, and for instructions on how to use the protocol with Sun ONE servers supporting Sun ONE Server Console administration.

Creating a Token

Directory Server uses a token and password to access the appropriate cryptographic key material on the accelerator board. The token takes the form *user@realm*, where *user* is a user in terms of the accelerator board — an owner of cryptographic keying material — and *realm* is a realm in terms of the accelerator board — a logical partition of users and their keying material. The accelerator board *user* need not bear any relation to a user account on the system. It is specific to the board. Refer to the accelerator board product documentation for further explanation of users and realms.

You may create a user and realm for the token using the `secadm`(1M) utility provided for use with the board. The accelerator board also permits creation of multiple *slots* to manage tokens for multiple applications. It is assumed here that for performance reasons, you dedicate the host to Directory Server and therefore use only one slot, the default. Refer to the accelerator board product documentation for details on using the board with multiple software applications.

Perform the following steps to create the user and realm for a token to access the default slot.

1. Start the `secadm` utility.

   ```
   $ CryptoPath/bin/secadm
   ```

   The default *CryptoPath* is `/opt/SUNWconn/crypto`.

2. Create a realm for the token.

   ```
   secadm> create realm=dsrealm
   System Administrator Login Required
   Login: super-user
   Password:
   Realm dsrealm created successfully.
   ```

3. Set the realm in which to create a user.

```
secadm> set realm=dsrealm
secadm{dsrealm}> su
System Administrator Login Required
Login: super-user
Password:
secadm{root@dsrealm}#
```

4. Create the user `nobody` to use the default slot, supplying the password used when restarting Directory Server with SSL configured.

```
secadm{root@dsrealm}# create user=nobody
Initial password: password
Confirm password: password
User nobody created successfully.
secadm{root@dsrealm}# exit
```

At this point you have created the user and realm for the token `nobody@dsrealm`, and supplied a password used when restarting Directory Server.

Generating Bindings for the Board

Bindings for the accelerator board take the form of an external security module you generate so Directory Server may bind to the board. Perform the following steps to generate a binding between the external security module and Directory Server certificate database with support for several SSL algorithms.

1. Set `LD_LIBRARY_PATH` before using `modutil`.

```
$ set LD_LIBRARY_PATH=ServerRoot/lib ; export LD_LIBRARY_PATH
```

2. Create a security module database if none exists.

```
$ cd ServerRoot/shared/bin
$ ./modutil -create -dbdir ../../alias -dbprefix "slapd-serverID"
```

3. Add the external security module to the security module database.

```
$ ./modutil -add "Crypto Mod" -dbdir ../../alias -nocertdb \
-libfile CryptoPath/lib/libpkcs11.so \
-mechanisms "RSA:DSA:RC4:DES" -dbprefix "slapd-serverID"
```

The default *CryptoPath* is `/opt/SUNWconn/crypto`.

4. List the security modules to ensure the add succeeded.

```
$ ./modutil -list -dbdir ../../alias -dbprefix "slapd-serverID"
```

You should see an entry for the `Crypto Mod` you added in Step 3.

5. Make the external security module the default for RSA, DSA, RC4, and DES.

   ```
   $ ./modutil -default "Crypto Mod" -dbdir ../../alias \
   -mechanisms "RSA:DSA:RC4:DES" -dbprefix "slapd-serverID"
   ```

   This should successfully change the default security module.

At this point you have generated bindings for the accelerator board and may import certificates.

Importing Certificates

Before configuring SSL, you must import the server and CA certificates you obtained as described in Table B-1 on page 165. Perform the following steps to import the certificates.

1. Import the server certificate `.p12` file.

   ```
   $ cd ServerRoot/shared/bin
   $ ./pk12util -i ServerCert.p12 -d ../../alias -P "slapd-serverID" \
   -h "nobody@dsrealm"
   Enter Password or Pin for "nobody@dsrealm": password
   Enter Password for PKCS12 file: password
   ```

2. Import the CA certificate.

   ```
   $ ./certutil -A -n "Crypto CA Cert" -t CT -i CACert.txt \
   -d ../../alias -P "slapd-serverID" -h "nobody@dsrealm"
   ```

3. List the certificates associated with the token to ensure the imports succeeded.

   ```
   $ ./certutil -L -d ../../alias -P "slapd-serverID" \
   -h "nobody@dsrealm"
   ```

   You should see entries for the certificates you added in Step 1 and Step 2.

At this point you have imported the certificates and may configure Directory Server to listen for SSL connections.

Configuring SSL

Using the token and password you created, bindings you generated between the external security module and Directory Server certificate database, and the certificates you imported, you may configure Directory Server to start in secure mode. Perform these steps to configure SSL and restart Directory Server in secure mode.

1. Create a file, `ssl.ldif`, of modifications to change SSL related Directory Server configuration entries.

Code Example B-1 Modifications to Activate SSL Using the Board (`ssl.ldif`)

```
dn: cn=RSA,cn=encryption,cn=config
changetype: add
objectclass: top
objectclass: nsEncryptionModule
cn: RSA
nsSSLToken: nobody@dsrealm
nsSSLPersonalitySSL: ServerCertNickname[1]
nsSSLActivation: on

dn: cn=encryption,cn=config
changetype: modify
replace: nsSSL3
nsSSL3: on
-
replace: nsSSLClientAuth
nsSSLClientAuth: allowed
-
replace: nsSSL3Ciphers
nsSSL3Ciphers: -rsa_null_md5,+rsa_rc4_128_md5,+rsa_rc4_40_md5,
 +rsa_rc2_40_md5,+rsa_des_sha,+rsa_fips_des_sha,+rsa_3des_sha,
 +rsa_fips_3des_sha,+fortezza,+fortezza_rc4_128_sha,
 +fortezza_null,+tls_rsa_export1024_with_rc4_56_sha,
 +tls_rsa_export1024_with_rc4_56_sha,
 +tls_rsa_export1024_with_des_cbc_sha
-
replace: nsCertfile
nsCertfile: alias/slapd-serverID-cert7.db
-
replace: nsKeyFile
nsKeyFile: alias/slapd-serverID-key3.db

dn: cn=config
changetype: modify
replace: nsslapd-secureport
nsslapd-secureport: port
-
replace: nsslapd-security
nsslapd-security: on
```

1. This nickname is contained in the certificate for Directory Server.

Here *port*, the value of `nsslapd-secureport`, is the port on which Directory Server listens for SSL connections once started in secure mode.

2. Apply the modifications to change Directory Server configuration.

```
$ ldapmodify -p currPort -D "cn=directory manager" -w password -f ssl.ldif
```

where *currPort* is the number of the port on which the Directory Server currently listens for client requests.

3. Restart the Directory Server in secure mode.

```
$ ServerRoot/slapd-serverID/restart-slapd
Enter PIN for nobody@dsrealm: password
```

Here *password* is the user password for `nobody` provided when the token `nobody@dsrealm` was created.

At this point, Directory Server listens for SSL traffic over the port you specified. You may configure Sun ONE Administration Server and client applications to access Directory Server over SSL through that port. Refer to the *Sun ONE Directory Server Administration Guide* for details.

Appendix C

Installing Sun Cluster HA for Directory Server

This appendix describes how to install and configure both the Sun Cluster HA for Directory Server data service and the associated Administration Server data service. Refer to the Sun Cluster 3.0 product documentation for Sun Cluster installation instructions and key concepts.

You must configure the data services as a failover services.

Before You Start

Use this section in conjunction with the worksheets in the *Sun Cluster 3.0 Release Notes* as a checklist before performing installation and configuration.

Prior to starting your installation, consider these questions.

- Do you plan to run multiple Directory Server instances on the same node?

  If so, you may choose to set `nsslapd-listenhost` on `cn=config` to the appropriate network resource (a logical host name, such as `dirserv.example.com`) as the IP address for each instance. Directory Server default behavior is to listen on all network interfaces.

- Do you run multiple data services in your Sun Cluster configuration?

  You may set up multiple data services in any order, with one exception: If you use Sun Cluster HA for DNS, you must set it up before setting up Sun Cluster HA for Directory Server.

Table C-1 summarizes the Sun Cluster HA for Directory Server installation and configuration process.

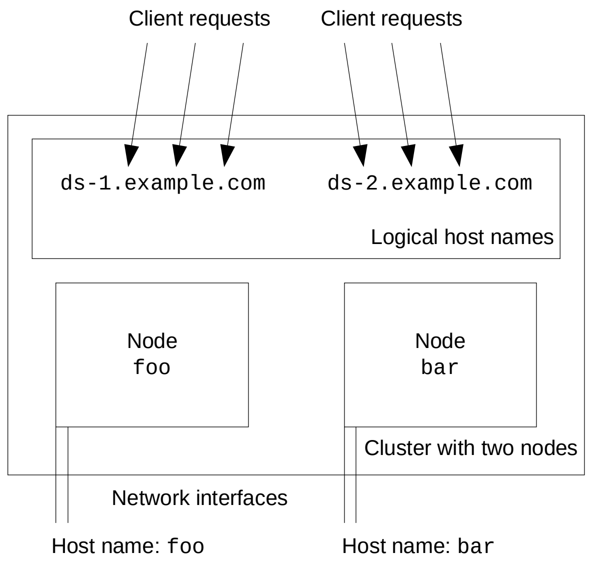
Table C-1 Installation and Configuration Process

Task	What you should know
"Setting Up Network Resources," on page 173	The names of the cluster nodes that can master the data services. The logical host names to be used by clients accessing Directory Server such as `ds1.example.com`, `ds2.example.com`. Refer to the Sun Cluster 3.0 product documentation for instructions on setting up logical host names.
"Installing the Servers," on page 175	The *ServerRoot* location on the global file system such as `/global/ds` where you install Directory Server. Installation details summarized in Table 1-2 on page 21.
"Installing the Data Service Packages," on page 176	The `SUNWdsha` and `SUNWasha` packages provide the management interface for the data services so you can manage Directory Server and Administration Server with the same tools as other data services in the cluster.
"Configuring the Servers," on page 176	The resource type names for Directory Server data service, `SUNW.dsldap`, and for the Administration Server data service, `SUNW.mps`. The names of the cluster nodes that can master the data services. The logical host names used by clients accessing Directory Server and Administration Server. The *ServerRoot* location on the global file system where you install Directory Server. The port on which Directory Server listens for client requests. The port on which Administration Server listens for client requests. The name of the resource group defined in "Setting Up Network Resources," on page 173.
"Configuring Extension Properties," on page 178	(Refer to the section itself for details.)

Setting Up Network Resources

Sun Cluster software manages logical host names that differ both from node names and from host names for individual network interfaces. Figure C-1 shows how logical host names, managed by a two-node cluster, are not permanently associated with either of the nodes.

Figure C-1 Cluster with Two Nodes



When installing the Sun Cluster HA for Directory Server data service, you configure Directory Server and Administration Server to listen on the logical host name interface so they are not tied to any particular node in the cluster, and the Sun Cluster software can manage failover. In Figure C-1, the nodes are named `foo` and `bar`. The logical host names you use during installation as shown in Figure C-1 however would be `ds-1.example.com` and `ds-2.example.com`, not `foo` and `bar`. Notice that the logical host names used are fully qualified domain names.

Refer to the Sun Cluster 3.0 product documentation for more information on these key concepts and for instructions on setting up logical host names.

After setting up logical host names, perform the following steps:

1. Become super user on a node in the cluster.

2. Verify that all network addresses you use have been added to the name service database.

   To avoid failures during name service lookup, ensure as well that all fully qualified domain names, fully qualified logical host names and shared IP addresses are present in the `/etc/hosts` file on each cluster node. Also configure name service mapping in `/etc/nsswitch.conf` on each cluster node to check local files first before trying to access other name services.

3. Create a failover resource group to hold network and application resources. For example:

   ```
   # scrgadm -a -g resource-group [-h node-list]
   ```

   Here *resource-group* specifies the name of the group.

   The optional *node-list* is a comma-separated list of physical node names or IDs identifying potential master nodes for the cluster. The order of the node names determines the order in which the nodes are considered as primary during failover. If all nodes in the cluster are potential masters, it is not necessary to specify the *node-list*.

4. Add logical host name resources to the resource group.

   ```
   # scrgadm -a -L -g resource-group -l logical-host-names [-n netif-list]
   ```

   Here *logical-host-names* is a comma-separated list of fully qualified domain names used as logical host names. You use one logical host name per Directory Server instance.

   The optional *netif-list* is a comma-separated list identifying the NAFO groups on each node. If you do not specify this option, `scrgadm`(1M) attempts to discover a network adapter on the subnet used by each logical host name specified on each node in *node-list* specified in Step 3.

5. Verify that all fully qualified domain names specified as logical host names in Step 4 have been added to the name service database.

6. Enable the resource group and bring it online.

   ```
   # scswitch -Z -g resource-group
   ```

With the resource group online, you may install the servers.

Installing the Servers

In Sun Cluster HA for Directory Server, both Directory Server and Administration Server run under the control of Sun Cluster. This means that instead of supplying the servers with a fully qualified domain name for the physical node during installation, you provide a fully qualified logical host name that can fail over to a different node.

You perform installation starting with the node online for the logical host name used by directory client applications, then repeating the process for all other cluster nodes that you want to master the Directory Server data service.

Installing on the Active Node

For the cluster node that is online for the logical host name used by directory client applications:

1. Install the Solaris packages for both Directory Server and Administration Server, referring to "Installing Solaris Packages," on page 24 for instructions.
2. Configure Directory Server. Refer to "Configuring Directory Server," on page 28 for instructions.

   When performing this step:

   - Place the Directory Server instance on the *global* cluster file system.
   - Use the logical host name, *not* the node name.
3. Configure Administration Server, referring to "Configuring Administration Server," on page 29 for instructions and using the same logical host name used to configure the Directory Server.
4. When using Directory Server in secure mode only, create an empty file named *ServerRoot*`/slapd-`*serverID*`/keypass` to indicate to the cluster that the Directory Server instance runs in secure mode.

   Also create a *ServerRoot*`/alias/slapd-`*serverID*`-pin.txt` file, containing the password required to start the instance automatically in secure mode. This allows the cluster to restart the data service without human intervention.

Installing on Other Nodes

For *each* node you want to master the Directory Server data service:

1. Install the Solaris packages for both Directory Server and Administration Server, referring to "Installing Solaris Packages," on page 24 for instructions.
2. Configure Directory Server using settings identical to those provided when "Installing on the Active Node," on page 175.
3. Configure Administration Server using settings identical to those provided when "Installing on the Active Node," on page 175.
4. Copy *ServerRoot*/`alias/slapd-`*serverID*`-pin.txt` from the first node to *ServerRoot*/`alias/`.

NOTE Do not remove or relocate any files placed on the global file system.

Installing the Data Service Packages

The data service packages, `SUNWdsha` and `SUNWasha`, provide the management interfaces for administering the servers as a data services within the cluster.

- On each cluster node that you want to support the Directory Server data service, use the `pkgadd`(1M) utility to install the data service packages.

```
# pkgadd -d dirContainingPackages SUNWasha SUNWdsha
```

Configuring the Servers

Perform the following steps *only* on the cluster node that is online for the logical host name in use by Directory Server:

1. Become super user.
2. Stop Directory Server and Administration Server.

```
# /usr/sbin/directoryserver stop
# /usr/sbin/mpsadmserver stop
```

3. Register the resource types for both data services.

```
# scrgadm -a -t SUNW.dsldap -f /etc/ds/v5.2/cluster/SUNW.dsldap
# scrgadm -a -t SUNW.mps -f /etc/mps/admin/v5.2/cluster/SUNW.mps
```

Here `SUNW.dsldap` and `SUNW.mps` are the predefined resource type names for the data services. `/etc/ds/v5.2/cluster/SUNW.dsldap` and `/etc/mps/admin/v5.2/cluster/SUNW.mps` define the data services.

4. Add the servers to the failover resource group created in “Setting Up Network Resources,” on page 173.

```
# scrgadm -a -j resource-name-ds -g resource-group -t SUNW.dsldap \
-y Network_resources_used=logical-host-name \
-y Port_list=port-number/tcp \
-x Confdir_list=ServerRoot/slapd-serverID

# scrgadm -a -j resource-name-as -g resource-group -t SUNW.mps \
-y Network_resources_used=logical-host-name \
-y Port_list=port-number/tcp \
-x Confdir_list=ServerRoot
```

Here you provide a new *resource-name-ds* to identify the Directory Server instance, and a new *resource-name-as* to identify the Administration Server instance.

The *resource-group* parameter is the name of the group specified in “Setting Up Network Resources,” on page 173.

The *logical-host-name* identifies the logical host name used for the current Directory Server instance.

The *port-number* is the numbers of the ports on which the server instances listen for client requests, specified in “Installing the Servers,” on page 175. Notice the `Port_list` parameter of each command takes only one entry.

ServerRoot and *ServerRoot*`/slapd-`*serverID* are paths specified in “Installing the Servers,” on page 175. Notice the `Confdir_list` parameter of each command takes only one entry.

5. Enable the server resources and monitors.

```
# scswitch -e -j resource-name-ds
# scswitch -e -j resource-name-as
```

Here *resource-name-ds* and *resource-name-as* are the names you provided to identify the servers in Step 4.

NOTE After configuring the servers, do not run backup and restore commands such as `db2bak`, `db2ldif`, `back2db`, and `ldif2db` on an inactive node of the cluster. Instead, perform all backup and restore procedures on the active node.

6. Consider performing the steps in the section, “Synchronizing HA Storage and Data Services,” on page 181 to improve performance on fail over.

Example Registration and Configuration

Code Example C-1 shows how you might register and configure the data service for the cluster illustrated in Figure C-1 on page 173.

Code Example C-1 Registering and Configuring the Data Service

```
(Create a failover resource group on the node that is online.)
# scrgadm -a -g ds-resource-group-1 -h foo,bar

(Add a logical hostname resource to the resource group.)
# scrgadm -a -L -g ds-resource-group-1 -l ds-1.example.com

(Bring the resource group online.)
# scswitch -Z -g ds-resource-group-1

(Install packages on each node in the cluster.)

(Stop the servers on the node that is online.)
# /usr/sbin/directoryserver stop
# /usr/sbin/mpsadminserver stop

(Register the SUNW.dsldap and SUNW.mps resource types.)
# scrgadm -a -t SUNW.dsldap -f /etc/ds/v5.2/cluster/SUNW.dsldap
# scrgadm -a -t SUNW.mps -f /etc/mps/admin/v5.2/cluster/SUNW.mps

(Create resources for the servers and add them to the resource group.)
# scrgadm -a -j ds-1 -g ds-resource-group-1 \
-t SUNW.dsldap -y Network_resources_used=ds-1.example.com \
-y Port_list=389/tcp \
-x Confdir_list=/global/ds/slapd-ds-1
# scrgadm -a -j as-1 -g ds-resource-group-1 \
-t SUNW.mps -y Network_resources_used=ds-1.example.com \
-y Port_list=5201/tcp \
-x Confdir_list=/global/ds

(Enable the application resources.)
# scswitch -e -j ds-1
# scswitch -e -j as-1
```

Configuring Extension Properties

Extension properties allow you to configure how the cluster software handles the application software. For example, you can adjust how the cluster determines when the data service must fail over.

What You Can Configure

You typically configure resource extension properties using the Cluster Module of the Sun Management Center, or using the `scrgadm` utility. You can change the extension properties listed in Table C-2 using the `scrgadm` utility with the `-x` *parameter=value* option.

Table C-2 `SUNW.dsldap` Resource Extension Properties

Property	Description	Default	Range
`Monitor_retry_count`	Integer value indicating the number of times the process monitor facility (PMF) restarts the fault monitor during the time window specified by the value of `Monitor_retry_interval`	4 attempts	-1 to 2,147,483,641 attempts -1 means retry forever.
`Monitor_retry_interval`	Integer value indicating the time in minutes over which failures of the fault monitor are counted If the number of times the fault monitor fails exceeds the value specified in `Monitor_retry_count` within this period, the PMF cannot restart the fault monitor.	2 minutes	-1 to 2,147,483,641 minutes -1 specifies an infinite retry interval.
`Probe_timeout`	Integer value indicating the timeout value in seconds that the fault monitor uses to probe a Directory Server instance	30 seconds	0 to 2,147,483,641 seconds

Refer to the Sun Cluster 3.0 product documentation for more information on Sun Cluster properties.

How the Fault Monitor Operates

The cluster software determines whether the data service is healthy using a fault monitor. The fault monitor probes the data service, and then determines whether the service is healthy or must be restarted based on the results of the probe.

Table C-3 How the Fault Monitor Interprets Probes

Directory Server running in...	Probe Used	Algorithm
Normal mode	`ldapsearch`	1. Attempt a search. 2. If the search operation results in: • `LDAP_SUCCESS`, then the service is considered healthy. • An LDAP error, then the service must be restarted. • A problem other than timeout, then the fault monitor probes again depending on `Monitor_retry_count` and `Monitor_retry_interval`. • The `Probe_timeout` duration being exceeded, then the fault monitor probes again depending on `Monitor_retry_count` and `Monitor_retry_interval`. Potential causes of timeout include heavy loads on the system, network, or Directory Server instance. Timeout may also indicate that the `Probe_timeout` value is set too low for the number of Directory Server instances monitored.
Secure mode (SSL)	TCP connect	1. Attempt to connect. 2. If the connection operation: • Succeeds, then the service is considered healthy. • Fails, then the service must be restarted. • Exceeds `Probe_timeout`, then the service must be restarted.

The fault monitor uses the IP addresses and port numbers you specified when “Configuring the Servers,” on page 176 to carry out probe operations. If Directory Server is configured to listen on two ports, one for SSL traffic and one for normal traffic, the fault monitor probes both ports using TCP connect, following the fault monitoring algorithm used for secure mode ports.

Synchronizing HA Storage and Data Services

The `SUNW.HAStorage` resource type synchronizes actions between HA storage and data services, permitting higher performance when a disk-intensive data service such as Directory Server undergoes fail over.

To synchronize a Directory Server data service with HA storage, complete the following steps on the node that is online for the logical host name in use by the data service:

1. Register the HA storage resource type.

   ```
   # scrgadm -a -t SUNW.HAStorage
   ```

2. Configure the storage resource to remain synchronized.

   ```
   # scrgadm -a -j HAStorage-resource-name -g HAStorage-resource-group \
   -t SUNW.HAStorage -x ServicePaths=volume-mount-point \
   -x AffinityOn=True
   ```

   Here, *volume-mount-point* identifies the disk volume where Directory Server stores data.

3. Enable the storage resource and monitors.

   ```
   # scswitch -e -j HAStorage-resource-name
   ```

4. Add a dependency on the existing Directory Server resource.

   ```
   # scrgadm -c -j resource-name-ds \
   -y Resource_Dependencies=HAStorage-resource-name
   ```

Refer to `SUNW.HAStorage`(5) for background information, and to the Sun Cluster 3.0 product documentation for further instructions on setting up a `SUNW.HAStorage` resource type for new resources.

Creating an Additional Directory Server Instance

Perform the following steps:

1. Create an additional Directory Server instance using the Sun ONE Server Console.

   Refer to the *Sun ONE Server Console Server Management Guide* for instructions.

2. Stop the new Directory Server instance on the node that is online for the logical host name in use by the data service.

   ```
   # /usr/sbin/directoryserver -server serverID stop
   ```

3. Add the Directory Server to the failover resource group created in “Setting Up Network Resources,” on page 173.

```
# scrgadm -a -j resource-name-ds -g resource-group -t SUNW.dsldap \
-y Network_resources_used=logical-host-name \
-y Port_list=port-number/tcp \
-x Confdir_list=ServerRoot/slapd-serverID
```

   Here you provide a new *resource-name-ds* to identify the Directory Server instance.

   The *resource-group* parameter is the name of the group specified in “Setting Up Network Resources,” on page 173.

   The *logical-host-name* identifies the logical host name used for the instance.

   The *port-number* is the number of the port on which the instance listens for client requests, specified in “Installing the Servers,” on page 175. Notice the `Port_list` parameter takes only one entry.

   ServerRoot and *ServerRoot*/`slapd-`*serverID* are paths specified in “Installing the Servers,” on page 175. Notice the `Confdir_list` parameter takes only one entry.

4. Enable the server resources and monitors.

```
# scswitch -e -j resource-name-ds
```

   Here *resource-name-ds* is the name you provided to identify the Directory Server in Step 3.

Uninstalling

To remove Sun Cluster HA for Directory Server and the associated Administration Server from the cluster, perform the following steps:

1. Stop the server instances.

```
# scswitch -n -j resource-name-ds
# scswitch -n -j resource-name-as
```

2. Remove the resources.

```
# scrgadm -r -j resource-name-ds
# scrgadm -r -j resource-name-as
```

3. Remove the resource types from the cluster database.

```
# scrgadm -r -t SUNW.dsldap
# scrgadm -r -t SUNW.mps
```

4. Delete the server configurations.

```
# /usr/sbin/mpsadmserver unconfigure
# /usr/sbin/directoryserver unconfigure
```

5. Remove the packages installed, including `SUNWdsha` and `SUNWasha`, from each node using the `pkgrm`(1M) utility.

Index