Getting Started Guide

Sun^{TM} ONE Directory Server

Version 5.2

816-6696-10
June 2003

SUNTONE℠
CERTIFIED

Contents

About This Guide

Sun™ ONE Directory Server 5.2 is a powerful and scalable distributed directory server 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 consolidates the information required by readers who are not familiar with directory service concepts, or with previous versions of Sun ONE Directory Server. This is not a reference manual but a comprehensive overview that will enable you to discover, install and evaluate Sun ONE Directory Server in a short period.

What's in This Guide

This guide includes the following information:

- *Documentation Overview* - describes the documentation set delivered with Sun ONE Directory Server 5.2 and indicates where you will find specific information. It also describes the major changes to the documentation set to assist users who are familiar with the documentation delivered in previous versions of the product

- *Introduction to Sun ONE Directory Server* - describes the basic concepts you must understand before designing and deploying your directory.
- *A Quick Look at Directory Server Console* - describes how to install Sun ONE Directory Server for evaluation / demonstration purposes, and how to use the console to examine the features described in the introduction.
- *A Quick Look at Directory Server Command-Line Utilities* - provides information on the `ldapsearch` and `ldapmodify` commands and describes how to use the command-line to examine the features described in the introduction.
- *Accessibility Features* - describes the accessible features of the Sun ONE Directory Server 5.2 user interface, including key mappings for the consoles.
- *Glossary* - a global glossary that defines the Sun ONE Directory Server terminology used throughout the documentation set.

Prerequisite Reading

Before reading this guide we strongly recommend you read the online release notes to obtain the latest information about new features and enhancements in this release of Sun ONE Directory Server. The release notes can be found at

```
http://docs.sun.com/doc/816-6703-10/index.html
```

Note that this guide does not attempt to provide comprehensive installation, reference or administrative information on Sun ONE Directory Server. For in depth information on these topics, please consult the relevant user guide in the Sun ONE Directory Server document set.

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 Installation	***ServerRoot* Path**
Solaris 9[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)

Table 1 Default *ServerRoot* Paths *(Continued)*

Product Installation	*ServerRoot* Path
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 Installation	Instance Location
Solaris 9	`/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 Sun ONE LDAP SDK for C and the Sun ONE LDAP 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/`

Chapter 1

Documentation Overview

This chapter describes the documentation set delivered with Sun ONE Directory Server 5.2 and indicates where you will find specific information. It also describes the major changes to the documentation set to assist users who are familiar with the documentation delivered in previous versions of the product.

Directory Server 5.2 Documentation Set

The Sun ONE Directory Server 5.2 documentation is provided in two separate editions. The documentation set you use will depend on whether or not you are using the multi-platform edition.

Both documentation sets includes the following user guides and reference manuals, delivered in HTML and PDF format:

- *Sun ONE Directory Server Getting Started Guide* - Consolidates all the information required by directory service novices, prospective purchasers of the product, and readers who are not completely familiar with basic LDAP concepts. This is not a reference manual but a comprehensive overview that will enable you to get up and running with Sun ONE Directory Server 5.2 in a short period.
- *Sun ONE Directory Server Deployment Guide* - Provides a foundation for planning your directory. This guide is intended for directory decision-makers, designers and administrators and should be your starting point if you have decided to use Sun ONE Directory Server and are in the process of planning your deployment. The guide includes a sample deployment scenario and several architectural strategies that indicate how Directory Server can be used to answer specific business requirements.

- *Sun ONE Directory Server Administration Guide* - Describes the procedures for managing directory contents and maintaining Directory Server. The Administration Guide includes procedures using the console interface and using the command line interface.
- *Sun ONE Directory Server Reference Manual* - Provides a reference for Directory Server configuration and the command-line utilities, and describes the standard schema for user directories provided with Sun ONE Directory Server 5.2. This manual combines the previous *Configuration, Command, and File Reference* and *Schema Reference* manuals and includes a description of the most significant error messages returned by Directory Server, along with their causes and suggestions on what to do, should they occur.
- *Sun ONE Directory Server Plug-In API Programming Guide* - Describes Directory Server plug-ins, libraries registered with Directory Server that customize and extend directory services provided by the product. This guide also indicates what has changed for this release of the plug-in API.
- *Sun ONE Directory Server Plug-In API Reference* - Describes the data definitions and functions available to server plug-in applications that customize and extend directory services provided by Sun ONE Directory Server 5.2. This reference manual does not cover earlier versions of the API, except to list deprecated functions.

In addition to the above documentation, the multi-platform documentation set includes:

- The Sun ONE Directory Server Release Notes in HTML format, that contain important information available at the time of the release of Sun ONE Directory Server 5.2. New features and enhancements, known limitations, and other late-breaking issues are addressed here. The latest version of the release notes is available online at http://docs.sun.com/doc/816-6703-10/index.html.
- *Sun ONE Directory Server Installation and Tuning Guide* - Provides information on how to install Sun ONE Directory Server 5.2. This guide replaces the previous Installation Guide and includes additional information on hardware sizing, operating system configuration, migration and upgrading, silent installation and uninstalling. It also provides guidance on how to tune Sun ONE Directory Server 5.2 for the best performance.

- *Sun ONE Directory Server Resource Kit (DSRK) Reference*. The Sun ONE DSRK provides tools and APIs for deploying, accessing, tuning, and maintaining the Sun ONE Directory Server. These utilities will help you implement and maintain more robust solutions based on LDAP, the Lightweight Directory Access Protocol. The LDAP SDKs (Software Development Kits) for C and Java™ programming languages make it easier to write client applications for your directory. These APIs expose all the functions for connecting to an LDAP directory and accessing or modifying its entries. Use them to design and integrate directory functionality into your applications at the programmatic level.
- *Sun ONE Server Console Server Managment Guide* **-** provides background information that system architects and administrators need to successfully install and manage Sun ONE servers in their enterprise.

A detailed description of the contents of the different user guides and reference manuals is provided in "Documentation Content," on page 16.

Documentation Content

This section provides a brief description of each document in the Sun ONE Directory Server documentation set.

Sun ONE Directory Server Getting Started Guide

This guide provides introductory information to the concepts of directory services in general and of Sun ONE Directory Server 5.2 in particular. It enables you to complete a basic installation of Sun ONE Directory Server and to perform the most essential administrative tasks, using the console and the command-line utilities, for evaluation purposes. This guide includes the following sections:

- Documentation Overview
- Introduction to Sun ONE Directory Server
- A Quick Look at Directory Server Console
- A Quick Look at Directory Server Command-Line Utilities
- Accessibility Features
- Glossary

Sun ONE Directory Server Deployment Guide

This guide provides you with a foundation for planning your directory. It includes sample deployment scenarios that illustrate how Sun ONE Directory Server 5.2 can be deployed to address a selection of business situations. The information provided here is primarily intended for directory decision-makers, solution designers, and administrators. This guide includes the following sections:

- Directory Server Design and Deployment Overview
- Planning and Accessing Directory Data
- Designing the Schema
- Designing the Directory Tree
- Designing the Directory Topology
- Designing the Replication Process
- Designing a Secure Directory

- Monitoring Your Directory
- A Sample Deployment Scenario
- Architectural Strategies
- Accessing Data Using DSML Over HTTP/SOAP

Sun ONE Directory Server Installation and Tuning Guide (multi-platform edition only)

This performance tuning guide provides accurate, reproducible recommendations and guidelines on how to correctly tune Sun ONE Directory Server 5.2 for optimal performance. This guide aims to outline the most important areas to be configured and tuned, in order to optimize Directory Server performance. This guide includes the following sections:

- Installing Sun ONE Directory Server
- Upgrading From Previous Versions
- Top Tuning Tips
- Hardware Sizing
- Tuning the Operating System
- Tuning Cache Sizes
- Tuning Indexing
- Tuning Logging
- Managing Use of Other Resources
- Installed Product Layout
- Using the Sun Crypto Accelerator Board
- Installing Sun Cluster HA for Directory Server

Sun ONE Directory Server Administration Guide

This guide describes all of the administration tasks you need to perform to maintain a directory service based on the Sun ONE Directory Server. It describes how to create directory entries and how to configure and populate directory databases, and covers access control and user account management. This guide includes the following sections:

- Introduction to Sun ONE Directory Server
- Creating Directory Entries
- Creating Your Directory Tree
- Populating Directory Contents
- Advanced Entry Management
- Managing Access Control
- User Account Management
- Managing Replication
- Extending the Directory Schema
- Managing Indexes
- Implementing Security
- Managing Log Files
- Monitoring Directory Server Using SNMP
- Using the Pass-Through Authentication Plug-In
- Using the UID Uniqueness Plug-In

Sun ONE Directory Server Reference Manual

This manual provides comprehensive reference information on the command-line utilities and scripts provided with Sun ONE Directory Server, configuration attributes, file formats, schemas, and error and connection codes. It also provides a reference of the information that is migrated when upgrading from previous versions of Directory Server. This manual includes the following sections:

- Command-Line Utilities
- Command-Line Scripts

- Core Server Configuration
- Core Server Configuration Attributes
- Plug-in Implemented Server Functionality
- Migration From Earlier Versions
- Server Instance Files
- Access Logs and Connection Codes
- About Schema
- Object Class Reference
- Attribute Reference
- Operational Attributes
- Error Codes
- ns-slapd and slapd.exe Command-Line Utilities
- Directory Internationalization
- LDAP URLs
- LDAP Data Interchange Format

Sun ONE Directory Server 5.2 Plug-In API Programming Guide

This guide shows you how to develop server plug-ins, libraries registered with Directory Server that customize and extend directory services offered as part of the product. This guide also indicates what has changed since the last release, so you can upgrade plug-ins written for previous versions of the product to function with the current version. This guide includes the following sections:

- Before You Start
- What's New
- Getting Started With Directory Server Plug-Ins
- Working With Entries
- Extending Client Request Handling
- Handling Authentication

- Performing Internal Operations
- Writing Entry Store and Entry Fetch Plug-Ins
- Writing Extended Operation Plug-Ins
- Writing Matching Rule Plug-Ins
- Writing Password Storage Scheme Plug-Ins

Sun ONE Directory Server Plug-In API Reference

This reference manual covers the data types and structures, functions and parameter block data that make up the public Sun ONE Directory Server plug-in API. Refer to it as you develop server plug-ins to extend Sun ONE Directory Server functionality. This manual includes the following sections:

- Data Type and Structure Reference
- Function Reference
- Parameter Block Reference

Sun ONE Directory Server Resource Kit Tools Reference

This reference manual covers the installation of the Sun ONE Directory Server Resource Kit (Sun ONE DSRK) and contains the command-line reference for all of its tools. This manual includes the following sections:

- Getting Started
- Directory Access Commands
- Performance Evaluation Tools
- LDIF Deployment Tools
- Maintenance and Debugging Tools
- Gateway Application
- Sun ONE LDAP Administrative Shell

Sun ONE Server Console Server Management Guide

This guide provides background information that system architects and administrators need to successfully install and manage Sun ONE servers in their enterprise. This guide includes the following sections:

- Overview of Sun ONE Server Console
- Sun ONE Server Console Basics
- Using Sun ONE Administration Server
- Advanced Server Management
- Public-Key Cryptography and SSL

Chapter 2

Introduction to Sun ONE Directory Server

Sun ONE Directory Server provides a central repository for storing and managing information. Almost any kind of information can be stored, from identity profiles and access privileges to information about application and network resources, printers, network devices and manufactured parts. Information stored in Sun ONE Directory Server can be used for the authentication and authorization of users to enable secure access to enterprise and Internet services and applications. Sun ONE Directory Server is extensible, can be integrated with existing systems, and enables the consolidation of employee, customer, supplier, and partner information.

This chapter describes the basic concepts you must understand before undertaking design and deployment strategies. It includes the following sections:

- What is a Directory Service?
- What is Sun ONE Directory Server?
- What's New in Sun ONE Directory Server 5.2

Note that this chapter does not attempt to explain in detail all the features of Sun ONE Directory Server. However, it provides you with enough information to start using Sun ONE Directory Server for evaluation purposes.

Once you have completed this chapter, you will be able to do the practical examples in the following two chapters, A Quick Look at Directory Server Console and A Quick Look at Directory Server Command-Line Utilities.

What is a Directory Service?

A directory service is the collection of software and processes that store information about your enterprise, subscribers, or both. In the context of this documentation, a directory service consists of at least one Directory Server and one or more directory client programs. Client programs can access names, phone numbers, addresses, and other data stored in the directory, depending on the permissions that have been set.

An example of a directory service is a Domain Name System (DNS) server. A DNS server maps a computer host name to an IP address. Thus, all of the computing resources (hosts) become clients of the DNS server. The mapping of host names enables users of the computing resources to locate computers on a network, using host names rather than complex numerical IP addresses.

The DNS server stores only two types of information: names and IP addresses. A directory service stores virtually unlimited types of information.

Sun ONE Directory Server stores all of these types of information in a single, network-accessible repository. The following are a few examples of the kinds of information you might store in a directory:

- Physical device information, such as data about the printers in your organization (where they reside, whether they are color or black and white, their manufacturer, date of purchase, serial number, IP address, and so forth).
- Public employee information, such as name, email address, and department.
- Contract or account information, such as the name of a client, final delivery date, bidding information, contract numbers, and project dates.
- Information on manufactured products, enabling manufacturing companies to locate and track their products more easily.
- Authentication information.

Sun ONE Directory Server serves the needs of a wide variety of applications. It also provides standard protocols and application programming interfaces (APIs) to access the information it contains.

The following sections describe global directory services, the Lightweight Directory Access Protocol (LDAP) and the Directory Services Markup Language (DSML).

About Global Directory Services

Sun ONE Directory Server provides global directory services, meaning it provides information to a wide variety of applications. Until recently, many applications came bundled with their own proprietary user databases, with information about the users specific to that application. While a proprietary database can be convenient if you use only one application, multiple databases become an administrative burden if the databases manage the same information.

For example, suppose your network supports three different proprietary email systems, each system with its own proprietary directory service. If users change their passwords in one directory, the changes are not automatically replicated in the others. Managing multiple instances of the same information results in increased hardware and personnel costs, a problem referred to as the *n + 1 directory problem*.

A global directory service solves the n+1 directory problem by providing a single, centralized repository of directory information that any application can access. However, giving a wide variety of applications access to the directory requires a network-based means of communicating between the applications and the directory. Sun ONE Directory Server provides two ways in which applications can access its global directory:

- Lightweight Directory Access Protocol (LDAP)
- Directory Services Markup Language (DSML)

About LDAP

LDAP provides a common language that client applications and servers use to communicate with one another. LDAP applications can easily search, add, delete and modify directory entries. LDAP is a “lightweight” version of the Directory Access Protocol (DAP) used by the ISO X.500 standard. DAP gives any application access to the directory via an extensible and robust information framework, but at an expensive administrative cost. DAP does not use the Internet standard TCP/IP protocol, has complicated directory-naming conventions, and generally has a high return on investment.

LDAP preserves the best features of DAP while reducing administrative costs. LDAP uses an open directory access protocol running over TCP/IP and uses simplified encoding methods. It retains the X.500 standard data model and can support millions of entries for a comparatively modest investment in hardware and network infrastructure.

About DSML

DSML is a markup language that enables you to represent directory entries and commands in XML. This means that XML-based applications using HTTP can take advantage of directory services while making full use of the existing web infrastructure. Sun ONE Directory Server 5.2 implements version 2 of the DSML standard (DSMLv2).

The Sun ONE Directory Server implementation of DSMLv2 differs slightly from the standard. For information on the restrictions in and extensions to the DSML standard, please refer to the *Sun ONE Directory Server Reference Manual*.

Directory Services and Databases

What is the difference between a directory service and a database? A database can be defined as an organized collection of data whose contents can easily be accessed, managed, and updated. Although a directory service can be considered an extension of a database, directory services generally have the following characteristics:

- *Hierarchical naming model*
  This naming model uniquely identifies a set of names so that there is no ambiguity when entries that have different origins but the same names are mixed together. Directory services operate in hierarchical namespaces, logically arranged in an inverse tree. A namespace is also referred to as a *suffix*.
- *Extended search capability*
  Directory services provide robust search capabilities, allowing searches on individual attributes of entries.
- *Distributed information model*
  A directory service enables directory data to be distributed across multiple servers within a network.
- *Shared network access*
  While databases are defined in terms of APIs, directories are defined in terms of protocols. Directory access implies network access by definition. Directories are designed specifically for *shared* access among applications. This is achieved through the object-oriented schema model. By contrast, most databases are designed for use only by particular applications and do not encourage data sharing.

- *Replicated data*
  Directories support replication (copies of directory data on more than one server) which make information systems more accessible and more resistant to failure.

- *Datastore optimized for reads*
  The storage mechanism in a directory service is generally designed to support a high ratio of reads to writes.

- *Extensible schema*
  The schema describes the type of data stored in the directory. Directory services generally support the extension of schema, meaning that new data types can be added to the directory.

What is Sun ONE Directory Server?

Sun ONE Directory Server includes the directory itself, the server-side software that implements the LDAP protocol, and a graphical user interface that allows users to search and change entries in the directory. Other LDAP clients are also available, including the directory managers in the Sun ONE Console. In addition, you can purchase other LDAP client programs or write your own using the LDAP client SDK included with the Sun ONE Directory Server product.

Without adding other client programs, Sun ONE Directory Server can provide the foundation for an intranet or extranet. Every Sun ONE server uses the directory as a central repository for shared server information, such as employee, customer, supplier, and partner data.

You can use Sun ONE Directory Server to manage extranet user-authentication, create access control, set up user preferences, and centralize user management. In hosted environments, partners, customers, and suppliers can manage their own areas of the directory, reducing administrative costs.

When you install Sun ONE Directory Server, the following components are installed on your machine:

- An LDAP server with a plug-in interface.
- Sun ONE Administration Server.
- Sun ONE Server Console to manage the servers.
- Command-line tools for starting and stopping the server, importing and exporting data in the database, database reindexing, account inactivation and deactivation, LDIF merges, kernel tuning, and replication management.

For more information about the command-line tools, refer to "A Quick Look at Directory Server Command-Line Utilities," on page 69 and to the tools information in the *Sun ONE Directory Server Reference Manual*.

- An SNMP agent

  For more information about SNMP monitoring, refer to the *Sun ONE Directory Server Administration Guide*.

- The Directory Services Markup Language (DSML).

Overview of Sun ONE Directory Server Architecture

At installation, Sun ONE Directory Server contains the following:

- Server front-ends responsible for network communications.
- Plug-ins for server functions, such as access control and replication.
- A basic directory tree containing server-related data.

The following sections describe each component of the directory in more detail.

Overview of the Server Front-Ends

The server front-ends of Sun ONE Directory Server manage communications with directory client programs. The Directory Server functions as a daemon. Multiple client programs can communicate with the server using LDAP over TCP/IP or DSML over HTTP. The connection can be protected using Secure Socket Layer over Transport Layer Security (SSL/TLS), depending on whether the client negotiates the use of TLS for the connection.

Communication that takes place with TLS, is usually encrypted. In the future, if DNS security is present, TLS used in conjunction with secured DNS will provide confirmation to client applications that they are binding to the correct server. If clients have been issued certificates, TLS can be used by Sun ONE Directory Server to confirm that the client has the right to access the server. TLS and its predecessor, SSL, are used throughout Sun ONE Directory Server products to perform other security activities such as message integrity checks, digital signatures, and mutual authentication between servers.

Multiple clients can bind to the server at the same time over the same network because the Sun ONE Directory Server is a multi-threaded application. As directory services grow to include larger numbers of entries or larger numbers of clients spread out geographically, they also include multiple Sun ONE Directory Servers placed in strategic places around the network.

Sun ONE Directory Server implements LDAP natively, thereby avoiding the performance and management overheads associated with having a gateway on top of an X.500 directory, and with relational databases.

Server Plug-ins Overview

Sun ONE Directory Server relies on plug-ins. A plug-in is a way to add functionality to the core server. For example, the *Uid Uniqueness* plug-in can be used to ensure that values given to the user id (`uid`) attribute are unique in the suffix configured when installing the directory.

A plug-in can be disabled. When disabled, the plug-in's configuration information remains in the directory but its function is not used by the server. Depending upon what you want your directory to do, you can choose to enable any of the plug-ins provided with Sun ONE Directory Server.

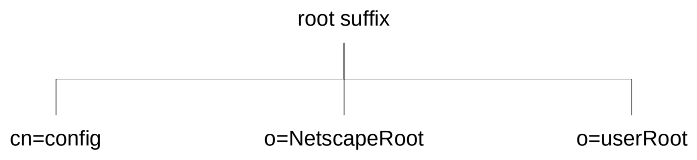
Sun ONE Professional Services can write custom plug-ins for any Sun ONE Directory Server deployment. Contact Sun ONE Professional Services for more information.

Overview of the Directory Tree

The directory tree, also known as a directory information tree or DIT, mirrors the tree model used by most file systems, with the tree's root, or first entry, appearing at the top of the hierarchy. At installation, Sun ONE Directory Server creates a default directory tree.

Figure 2-1 represents the structure of the default directory tree:

Figure 2-1 Default Directory Tree



The root of the tree is called the root suffix.

At installation, the directory contains three subtrees under the root suffix:

- `cn=config`

  where `cn` stands for Common Name. This subtree contains information about the server's internal configuration.

- `o=NetscapeRoot`

  where `o` stands for Organization. This subtree contains the configuration information of other Sun ONE servers, such as Sun ONE Administration Server. The Administration Server takes care of authentication and all actions that cannot be performed through LDAP (such as starting or stopping Directory Server). This subtree name originates from a legacy version of the product.

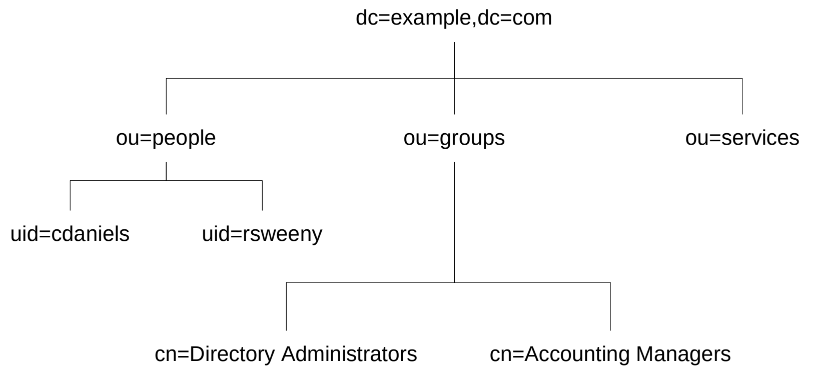
- `o=userRoot`

  During installation, a user database is created by default. The default name of the user database is `o=userRoot`. You can choose to populate this database at installation, or to populate it later.

NOTE	When you install another instance of Directory Server, you can specify that it does not contain the `o=NetscapeRoot` information, but that it uses the configuration directory (or the `o=NetscapeRoot` subtree) present on another server. See the *Sun ONE Server Console Server Management Guide* for more information about deciding upon the location of your configuration directory.

You can build on the default directory tree to add any data relevant to your directory installation. In Figure 2-2, the `o=userRoot` suffix has been renamed to `dc=example,dc=com`, and additional subtrees have been added to reflect the organizational hierarchy.

Figure 2-2 Sample Directory Tree



In the preceding figure:

- `dc` refers to the domain component
- `ou` refers to the organizational unit
- `uid` refers to the user id

Directory Server Data Storage

Directory data is stored in an internal database that is implemented as a plug-in. The database plug-in is automatically installed with the directory and is enabled by default.

By default, Sun ONE Directory Server uses a single database to store the directory tree. This database can manage millions of entries. The default database supports advanced methods of backing up and restoring data, so that the data is not at risk.

You can use multiple databases to support Directory Server. You can also distribute data across the databases, enabling the server to store more data than can be held in a single database.

The following sections describe how a directory database stores data.

About Directory Entries

LDAP Data Interchange Format (LDIF) is a standard text-based format for describing directory entries. An entry is a group of lines in an LDIF file that contains information about an object, such as a person in your organization or a printer on your network. Information about the entry is represented in the LDIF file by a set of attributes and their values. Each entry has an object class attribute that specifies the kind of object the entry describes and defines the set of additional attributes it contains. Each attribute describes a particular trait of an entry.

For example, an entry might have the object class `organizationalPerson`, indicating that the entry represents a person within a particular organization. This object class allows the `givenname` and `telephoneNumber` attributes. The values assigned to these attributes give the name and phone number of the person represented by the entry.

Sun ONE Directory Server also uses read-only attributes that are calculated by the server. These attributes are called *operational attributes*. There are also some operational attributes that can be set by the administrator, for access control and other server functions.

Entries are stored in a hierarchical structure in the directory tree. In LDAP, you can query an entry and request all entries below it in the directory tree. This subtree is called the base distinguished name, or base DN. For example, if you make an LDAP search request specifying a base DN of `ou=people,dc=example,dc=com`, the search operation examines only the `ou=people` subtree in the `dc=example,dc=com` directory tree.

Note that not all entries are automatically returned in response to an LDAP search. Entries of the `ldapsubentry` object class are not returned in response to normal search requests. An `ldapsubentry` entry represents an administrative object, for example the entries that are used internally by Directory Server to define a role or a class of service. To receive these entries, clients must search specifically for entries of the `ldapsubentry` object class.

The LDIF format is described in detail in the *Sun ONE Directory Server Reference Manual*.

Distributing Directory Data

When you store various parts of a tree in separate databases, your directory can process client requests in parallel, improving performance. You can also store databases on different machines, to improve performance further.

To connect distributed data, you can create a special entry in a subtree of your directory. All LDAP operations attempted below this entry are sent to a remote machine where the entry is actually stored. This method is called *chaining*.

Chaining is implemented in the server as a plug-in, which is enabled by default. Using this plug-in, you create database links (special entries that point to data stored remotely). When a client application requests data from a database link, the database link retrieves the data from the remote database and returns it to the client.

Managing Data in Directory Server

The database is the basic unit of storage, performance, replication, and indexing. A variety of operations can be performed on a database, including importing, exporting, backing up, restoring, and indexing.

Importing Data

Sun ONE Directory Server provides three methods for importing data:

- Importing from the Directory Server Console.

  You can use the Directory Server Console to append data to all of your databases, including database links.

- Initializing databases.

  You can use the Directory Server Console to import data to one database. This method overwrites any data contained by the database.

- Importing data from the command line.

  You can import data using the command-line utilities `ldif2db`, `ldif2db.pl`, and `ldif2ldap`. These utilities are described in more detail in Chapter 4, “A Quick Look at Directory Server Command-Line Utilities.”

Exporting Data

You can use LDIF to export database entries from your databases. LDIF is a standard format described in RFC 2849, "The LDAP Data Interchange Format (LDIF) - Technical Specification."

Exporting data can be useful for the following:

- Backing up the data in your database
- Copying your data to another Directory Server
- Exporting your data to another application
- Repopulating databases after a change to your directory topology

You can use the Directory Server console or the command-line utilities `db2ldif` and `db2ldif.pl` to export data.

Backing Up and Restoring Data

You can use Directory Server Console or the `db2bak` command line utility to back up directory data. Both methods allow you to perform a backup while the server is running, which prevents you having a period during which the directory is not accessible.

You can restore data from a previously generated backup using Directory Server Console or the command-line utilities `bak2db` and `bak2db.pl`. Restoring databases overwrites any existing database files. While restoring databases, the server must be running. However, the databases are unavailable for processing operations during the restore.

Indexing Data

Depending on the size of your databases, searches performed by client applications can take a lot of time and resources. You can use indexes to improve search performance.

Indexes are files stored in the directory databases. Separate index files are maintained for each database in the directory. Each file is named according to the attribute it indexes. The index file for a particular attribute can contain multiple types of indexes, allowing you to maintain several types of index for each attribute. For example, a file called `givenName.db3` contains all the indexes for the `givenName` attribute.

Depending on the types of applications using your directory, you will use different types of index. Different applications may frequently search for a particular attribute, or may search your directory in a different language, or may require data in a particular format.

Sun ONE Directory Server supports the following types of index:

- *Presence index*

  The presence index lists entries that possess a particular attribute, such as `uid`.

- *Equality index*

  The equality index lists entries that contain a specific attribute value, such as `cn=Charlene Daniels`.

- *Approximate index*

The approximate index allows approximate (or "sounds-like") searches. For example, an entry contains the attribute value of `cn=Charlene L. Daniels`. An approximate search would return this value for searches against `cn~=Charlene Daniels`, `cn~=Charlene`, and `cn~=Daniels`.

Note that approximate indexes work only for English language entries, in ASCII characters.

- *Substring index*

  The substring index allows searches against substrings within entries. For example, a search for `cn=*derson` would match common names containing this string (such as Bill Anderson, Norma Henderson, and Steve Sanderson).

- *International index*

  Associates the object identifier (OID) of a locale with the attributes to be indexed to speed up searches in international directories.

- *Browsing index*

  The browsing, or virtual list view (VLV), index speeds up the display of entries in Directory Server Console. You can create a browsing index on any branch in your directory tree to improve the display performance.

Directory Server Schema

Directory schema maintains the integrity of the data stored in your directory by imposing constraints on the size, range, and format of data values. You decide what types of entries your directory contains (people, devices, organizations, and so forth) and the attributes available to each entry.

The predefined schema included with Directory Server contains both the standard LDAP schema as well as additional application-specific schema to support the features of the server. While this schema meets most directory needs, you may need to extend it with new object classes and attributes to accommodate the unique needs of your directory. Refer to the *Sun ONE Directory Server Deployment Guide* for information on extending the schema.

The following sections describe the format, standard attributes, and object classes included in the Sun ONE standard schema.

Schema Format

Directory Server bases its schema format on version 3 of the LDAP protocol (LDAPv3). This protocol requires directory servers to publish their schemas through LDAP itself, allowing directory client applications to retrieve the schema and adapt their behavior based on it. The global set of schema for Directory Server can be found in the entry named `cn=schema`.

Directory Server schema differs slightly from the LDAPv3 schema, as it uses its own proprietary object classes and attributes. In addition, it uses a private field in the schema entries called X-ORIGIN, which describes where the schema entry was defined originally. For example, if a schema entry is defined in the standard LDAPv3 schema, the X-ORIGIN field refers to RFC 2252. If the entry is defined by Sun Microsystems, Inc. for Directory Server's use, the `X-ORIGIN` field contains the value `Sun ONE Directory Server`.

For example, the standard `person` object class appears in the schema as follows:

```
objectClasses: ( 2.5.6.6 NAME 'person' DESC 'Standard LDAP
objectclass' SUP top MUST ( sn $ cn ) MAY ( description $ seeAlso
$ telephoneNumber $ userPassword ) X-ORIGIN 'RFC 2256' )
```

This schema entry states the object identifier, or OID, for the class (`2.5.6.6`), the name of the object class (`person`), a description of the class (`Standard Person Object Class`), then lists the required attributes (`objectclass`, `sn`, and `cn`) and the allowed attributes (`description`, `seealso`, `telephoneNumber`, and `userPassword`).

Standard Attributes

Attributes hold specific data elements such as a name or a fax number. Directory Server represents data as attribute-data pairs, a descriptive attribute associated with a specific piece of information. For example, the directory can store a piece of data such as a person’s name in a pair with the standard attribute, in this case `commonName` (`cn`). So, an entry for a person named Charlene Daniels has the following attribute-data pair:

```
cn: Charlene Daniels
```

In fact, the entire entry is represented as a series of attribute-data pairs. The entire entry for Charlene Daniels might appear as follows:

```
dn: uid=cdaniels, ou=people, dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Charlene Daniels
```

```
sn: Daniels
givenName: Charlene
givenName: Charlie
mail: cdaniels@example.com
```

Notice that the entry for Charlene contains multiple values for some of the attributes. The attribute `givenName` appears twice, each time with a unique value. The object classes that appear in this example are explained in the next section, “Standard Object Classes."

In the schema, each attribute definition contains the following information:

- A unique name
- An object identifier (OID) for the attribute
- A text description of the attribute
- The OID of the attribute syntax
- Indications of whether the attribute is single-valued or multi-valued, whether the attribute is for the directory’s own use, the origin of the attribute, and any additional matching rules associated with the attribute.

For example, the `cn` attribute definition appears in the schema as follows:

```
attributetypes: ( 2.5.4.3 NAME 'cn' DESC 'commonName Standard
Attribute' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The SYNTAX is the OID of the syntax for values of the attribute. For a complete description of the attribute syntax definitions, see “About Schema” in the *Sun ONE Directory Server Reference Manual*. For more information about the LDAPv3 schema format, refer to the LDAPv3 Attribute Syntax Definitions document (RFC 2252).

Standard Object Classes

Object classes are used to group related information. Typically, an object class represents a real object, such as a person or a fax machine. Before you can use an object class and its attributes in your directory, it must be identified in the schema. The directory recognizes a standard list of object classes by default. For more information, refer to Chapter 10, “Object Class Reference” in the *Sun ONE Directory Server Reference Manual*.

Each directory entry belongs to one or more object classes. Once you place an object class identified in your schema on an entry, you are telling Directory Server that the entry can have a certain set of attribute values and must have another, usually smaller, set of attribute values.

Object class definitions contain the following information:

- A unique name
- An object identifier (OID) that names the object
- A set of mandatory attributes
- A set of allowed attributes

For an example of a standard object class as it appears in the schema, refer to "Schema Format," on page 36.

Groups, Roles and Classes of Service

Information in the directory is organized hierarchically. This hierarchy is a grouping mechanism, although it is not well suited for associations between dispersed entries, for frequently changing organizations, or for data that is repeated in many entries.

As a solution to, groups and roles provide more flexible associations between entries, and class of service simplifies the management of data that is shared within branches of your directory.

Static and Dynamic Groups

A group is an entry that specifies the other entries that are its members. When you know the name of a group, it is easy to retrieve all of its member entries.

- Static groups explicitly name their member entries. Static groups are suitable for groups with few members, such as the group of directory administrators.
- Dynamic groups specify a filter, and all entries that match are members of the group. These groups are dynamic because membership is defined every time the filter is evaluated.

The advantage of groups is that they make it easy to find all of their members. Static groups may simply be enumerated, and the filters in dynamic groups may simply be evaluated. The disadvantage of groups is that given an arbitrary entry, it is difficult to name all the groups of which it is a member.

Managed, Filtered and Nested Roles

Roles are an alternative entry grouping mechanism that automatically identifies all roles of which any entry is a member. When you retrieve an entry in the directory, you immediately know the roles to which it belongs. This overcomes the main disadvantage of the group mechanism.

- Managed roles are the equivalent of static groups, except that membership is defined in each member entry and not in the role definition entry.
- Filtered roles are similar to dynamic groups. They define a filter that determines the members of the role.
- Nested roles name other role definitions, including other nested roles. The set of members of a nested role is the union of all members of the roles it contains. Nested roles may also define extended scope to include the members of roles in other subtrees.

Class of Service

The class of service (CoS) mechanism allows attributes to be shared between entries in a way that is invisible to applications. CoS does not define membership but rather allows related entries to share data for coherence and space considerations.

For example, a directory may contain thousands of entries that all have the same value for the `facsimileTelephoneNumber` attribute. Traditionally, to change the fax number, you would need to update each entry individually, a large job for administrators that runs the risk of not updating all entries. Using CoS, the fax number is stored in a single place, and the directory server automatically generates the `facsimileTelephoneNumber` attribute on every concerned entry as it is returned.

To client applications, a generated CoS attribute is retrieved just as any other attribute. However, directory administrators now have only a single fax value to manage. In addition, because there are less values actually stored in the directory, the database uses less disk space. In general, a stored attribute value will take precedence over a CoS generated value for the same attribute. However, the CoS mechanism can also override stored values, or generate multiple values for the same attribute.

Security in Directory Server

Sun ONE Directory Server provides the following security methods:

- Authentication

  A means whereby one party verifies another's identity. For example, a client gives a password to Directory Server during a bind operation (the first request the server receives from a client.)

- Password policy

Defines the criteria that a password must satisfy to be considered valid, for example, age, length, and syntax.

- Encryption

  Protects the privacy of information. When data is encrypted, it is converted into a form that only the intended recipient can understand.

- Access control

  Controls the access rights granted to different directory users, and provides a means of specifying required credentials or bind attributes.

- Account inactivation

  Disables a user account, group of accounts or an entire domain so that all authentication attempts are automatically rejected.

- Signing with SSL

  Maintains the integrity of information. If information is signed, the recipient can determine that it was not tampered with during transit.

- Auditing

  Allows you to determine if the security of your directory has been compromised. For example, you can audit the log files maintained by your directory.

These tools for maintaining security can be used in combination in your security design. You can also use other features of the directory such as replication and data distribution to support your security design.

Replication in Directory Server

Replication is the mechanism that automatically copies directory data from one Directory Server to another. Using replication, you can copy any directory tree or subtree (stored in its own database), or specific attributes of entries between servers. The Directory Server that holds the master copy of the information, automatically copies updates to the other servers.

Replication enables you to provide a highly available directory service, and to geographically distribute your data. In practical terms, replication brings the following benefits:

- Fault tolerance/Failover

By replicating directory trees to multiple servers, you can ensure your directory is available even if some hardware, software, or network problem prevents your directory client applications from accessing a particular Directory Server. Your clients are referred to another directory server for read and write operations. Note that to support write failover you must have a multi-master replication environment (see the definition of *Masters* on page 41 for more information).

- Load balancing

  By replicating your directory tree across servers, you can reduce the access load on any given machine, thereby improving server response time.

- Higher performance and reduced response times

  By replicating directory entries to a location close to your users, you can vastly improve directory response times.

- Local data management

  Replication allows you to own and manage data locally while sharing it with other Directory Servers across your organization.

Replication Concepts

Replica

A replica refers to a database that participates in replication. A replica can take on one of the following roles:

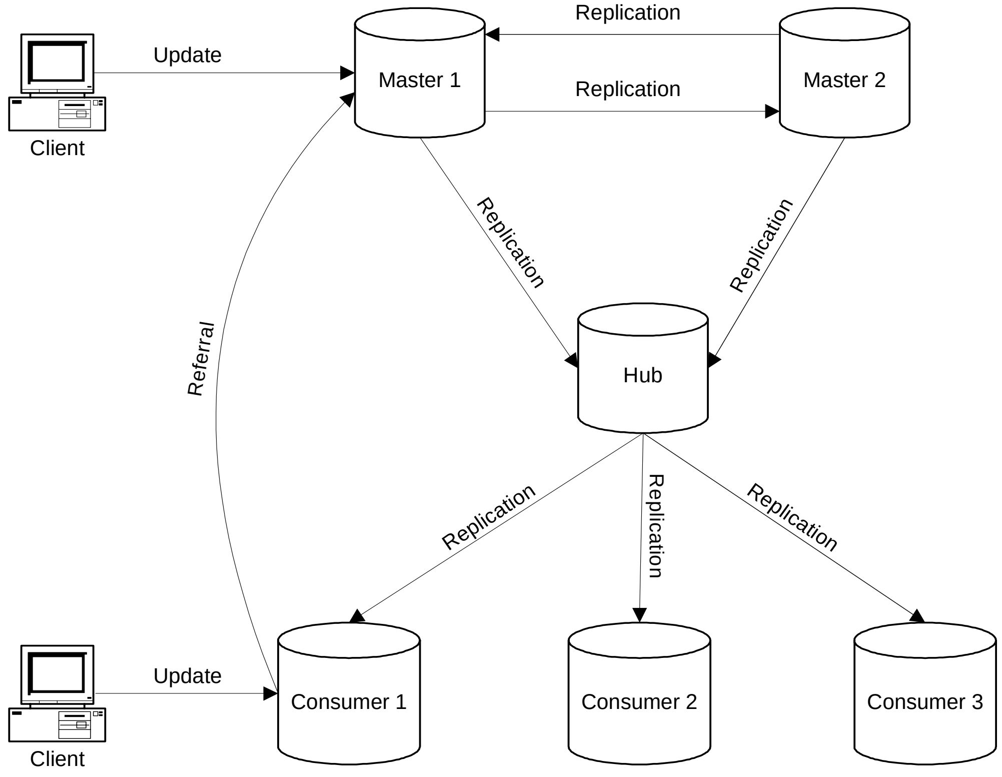
- *Master*
  A master is a read-write database that contains a master copy of the directory data and accepts update requests from directory clients. A master replica can be one of a set of multiple masters, or a single master. Multi-masters accept replication from other masters. Single masters do not accept replication from other replicas.

- *Dedicated Consumer*
  A dedicated consumer is a read-only database that contains a copy of the information held in the master replica. Thus, a consumer replica accepts replication from one or more masters. A consumer replica can process search requests from directory clients but refers update requests to the master replica. This is known as *referral*.

- *Hub*
  A hub is also a read-only database, like a consumer replica. A hub accepts replication from one or more masters but is also able to replicate to consumers. A hub does not accept client updates.

The following figure shows a basic multi-master replication scenario, indicating the three different roles that a replica can have and the processes that occur between replicas.

Figure 2-3 Basic Multi-Master Replication Scenario



Supplier Servers and Consumer Servers

In any replication scenario, the replica sending updates is called a supplier and the server that receives updates is called a consumer. Therefore, a server that manages a master replica that it replicates to other servers is called a supplier server or master server. A server that manages a consumer replica that is updated by a different server is called a consumer server.

Note that a server can be both a supplier and a consumer. This is true in the following cases:

- When Directory Server manages a combination of master replicas and consumer replicas.
- When Directory Server acts as a hub supplier, that is, it receives updates from a master server and replicates the changes to consumer servers. This is known as *cascading replication*.
- When a master replica is mastered on two different Directory Servers, each Directory Server acts as a supplier and a consumer of the other Directory Server. This is known as *multi-master replication*.

In Sun ONE Directory Server 5.2, replication is always initiated by the supplier server, never by the consumer server. In other words, replication is *supplier-initiated*. In setting up replication, you configure one or more supplier servers to push data to one or more consumer servers.

For any particular replica, the supplier server must:

- Respond to read, add and modify requests from directory clients.
- Maintain state information and a change log for the replica.
- Initiate replication to consumer servers.

The supplier server is always responsible for recording the changes made to the supplier replicas that it manages. It makes sure that any changes are replicated to consumer servers.

A consumer server must:

- Respond to read requests.
- Refer add and modify requests to the supplier server for the replica.

Any time a request to add, delete, or change an entry is received by a consumer server, the request is referred to the supplier for the replica. The supplier server performs the request, then replicates the change.

In the special case of cascading replication, the hub supplier must:

- Respond to read requests.
- Refer add and modify requests to the supplier server for the replica.
- Initiate replication to consumer servers.

Change Log

Each supplier server maintains a change log. A change log is a record that describes the modifications that have occurred on a supplier replica. The supplier server replays these modifications to the replicas stored on consumer servers, or to other masters in the case of multi-master replication.

When an entry is modified, added or deleted, a change record describing the LDAP operation that was performed is recorded in the change log.

Unit of Replication

In Sun ONE Directory Server 5.2, the smallest unit of replication is a database. The replication mechanism also requires that one database correspond to one suffix. You cannot replicate a suffix that is distributed over two or more databases.

Fractional Replication

Fractional replication is a new feature in Sun ONE Directory Server 5.2 and refers to the ability to replicate only certain attributes of an entry. In fractional replication, all entries in a database are replicated, but certain attributes of these entries may be filtered out. For example, a bank may wish to replicate all the details of its customers' accounts, other than their credit card numbers. Using fractional replication, the bank can exclude only the attribute that relates to the credit card number from the data that is replicated.

Replication Agreement

Directory Server uses replication agreements to define replication. A replication agreement describes replication between one supplier and one consumer. The agreement is configured on the supplier server and identifies the following:

- The database to replicate.
- The consumer server to which the data is pushed.
- The times during which replication can occur.
- The DN and credentials the supplier server must use to bind to the consumer, called the Replication Manager entry or supplier bind DN.
- How the connection is secured (SSL, client authentication).

Replication Identity

When replication occurs between two servers, the consumer server authenticates the supplier when it binds to send replication updates. This authentication process requires that the entry used by the supplier to bind to the consumer is stored on the consumer server. This entry is called the Replication Manager entry, or supplier bind DN.

The Replication Manager entry, or any entry you create to fulfill that role, must meet the following criteria:

- There must be at least one on every server that manages consumer replicas (or hub replicas).
- This entry must not be part of the replicated data for security reasons.

NOTE This entry has a special user profile that bypasses all access control rules defined on the consumer server.

When you configure replication between two servers, you must identify the Replication Manager (supplier bind DN) on both servers:

- On the consumer server or hub supplier, when you configure the consumer replica or hub replica, you must specify this entry as the one authorized to perform replication updates.
- On the supplier server, when you configure the replication agreement, you must specify the DN of this entry in the replication agreement.

NOTE In the Directory Server Console, the Replication Manager entry is referred to as the *supplier bind DN*, which may be misleading as the entry does not exist on the supplier server. It is called the supplier bind DN because it is the entry that must be present on the consumer so that it can authenticate the supplier when it binds to provide replication updates to the consumer.

What’s New in Sun ONE Directory Server 5.2

Sun ONE Directory Server 5.2 contains the following new features and enhancements:

- Updated and improved server management console

  Directory Server Console now offers a simplified interface for configuring replication and provides support for IPv6. For details on the new console, refer to “Using the Directory Server Console” in the *Sun ONE Directory Server Administration Guide.*

- Enhanced replication functionality

  New replication features include:

  - Support for 4-way multi-master replication
  - Support for multi-master replication over WAN
  - Online promotion and demotion of servers
  - Fractional replication (the ability to replicate a subset of attributes)
  - The following replication monitoring tools:

    `insync` - indicates the synchronization state between a master replica and one or more consumer replicas.

    `entrycmp` - compares the attributes and values of the same entry on two different servers.

    `repldisc` - enables you to discover a replication topology, constructing a graph of all known servers and displaying a matrix describing the topology.

    Note that these tools are also compatible with Directory Server 5.1 Service Packs 1 and 2.

  - Improved replication failover
  - Improved concurrent replication updates

    Concurrent changes can be received on a single consumer from multiple suppliers.

    Within a replication session, multiple updates can be replayed concurrently.

- Directory access through DSMLv2

- Support for IPv6
- Large cache support (64-bit)
- Ability for LDAP clients to obtain their effective access rights
- A simplified migration process from version 4.x and 5.x to version 5.2
- Multiple password policies
- An interactive GUI installer
- Support for startTLS on Windows
- Improved error logging
- Flexible role scope
- Ability to use virtual attributes in search filters
- The ability to encrypt attributes other than passwords
- Support for Sun Crypto Accelerator 1000 Board
- Performance improvements
- Advanced binary copy

  Enables the cloning of master or consumer replicas using the binary backup files from one server to restore the identical directory contents on another server.

These new features are documented in the *Sun ONE Directory Server Administration Guide* and the *Sun ONE Directory Server Deployment Guide*.

Chapter 3

A Quick Look at Directory Server Console

This chapter provides practical examples (using Directory Server Console) of the features described in the previous chapter. It walks you through the essential tasks you need to perform to have an overview of how Directory Server works. Information on how the command-line utilities are used to perform these tasks is provided in the next chapter.

Note that this chapter does not attempt to provide comprehensive installation or administration information. Detailed installation and configuration procedures are described in the *Sun ONE Directory Server Installation and Tuning Guide* and in the *Solaris System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*. Detailed administration procedures are described in the *Sun ONE Directory Server Administration Guide*.

This chapter includes the following sections:

- Installing and Configuring Directory Server
- Using Directory Server Console
- Managing Entries
- Understanding the Schema
- Working with Groups and Roles
- Working With Class of Service (CoS)
- Working With ACIs
- Setting Up Replication

Installing and Configuring Directory Server

The method used to install and configure Sun ONE Directory Server will differ, depending on whether the software is bundled with your operating system. For detailed instructions on how to install Directory Server unbundled version, refer to the *Sun ONE Directory Server Installation and Tuning Guide*. For instructions on how to configure the bundled version, refer to the *Solaris System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

Before You Start

In configuring Sun ONE Directory Server, you will provide certain information. The examples that follow assume the information as summarized in the following table. If you supply different variables in the installation, make a note of these so that you can use them in the examples.

Table 3-1 Basic Information Required During Typical Installation

Description	Examples
Administration domain	`example.com`
Administration Server port number	`5200`
Directory Administrator ID	`admin`
(The Directory Administrator is a standard user created in the configuration directory, with access to the Sun ONE Server Console.)	
Directory Administrator password	`admin`
Directory Manager ID	`cn=Directory Manager`
(The Directory Manager is a special user with full access rights to the directory.)	
Directory Manager password	`password`
Directory Server port number	`5201`
Server ID	`myServer`
Server suffix	`dc=example,dc=com`
Server user ID	Install Directory Server as yourself, on all platforms except Windows, where you need administrative privileges.
ServerRoot (software installation directory)	`/var/Sun/mps`

When Directory Server is minimally configured and started, you are ready to proceed with the examples in this guide. The installation program indicates the host and port numbers used by the Administration Server.

Using Directory Server Console

You perform most Directory Server administrative tasks through the Administration Server, a second server provided to help you manage Directory Server (and all other Sun ONE servers). Sun ONE Server Console is the graphical interface to the Administration Server. Directory Server Console is a part of Sun ONE Server Console, and is designed specifically for use with Sun ONE Directory Server.

You can perform most Directory Server administrative tasks from the Directory Server Console. You can also perform administrative tasks manually by editing the configuration files or by using command-line utilities. The command-line utilities are discussed in the next chapter.

NOTE All the commands described in this section can be accessed via the `directoryserver` command in the Solaris packaged version. For more information, refer to the `directoryserver(1)` man page.

Starting Directory Server Console

To start Directory Server Console:

1. Check that the directory server daemon, `slapd-`*serverID* is running (using the `ps` command, or an equivalent command for your operating system.) If it is not, enter the following command to start it:

   ```
   % ServerRoot/slapd-serverID/start-slapd
   ```

2. Check that the administration server daemon, `admin-serv` is running (using the `ps` command, or an equivalent command for your operating system.) If it is not, enter the following command to start it:

   ```
   % ServerRoot/start-admin
   ```

3. Start Sun ONE Server Console by entering the following command

   ```
   % ServerRoot/startconsole
   ```

The Console login window is displayed. If your configuration directory (the directory that contains the `o=NetscapeRoot` suffix) is stored in a separate instance of Directory Server, a window is displayed requesting the administrator user DN, password, and the URL of the Admin Server for that directory server.

4. Log in using the bind DN and password of a user with sufficient access permissions for the operations you want to perform. For example, use `cn=Directory Manager`, and the appropriate password.

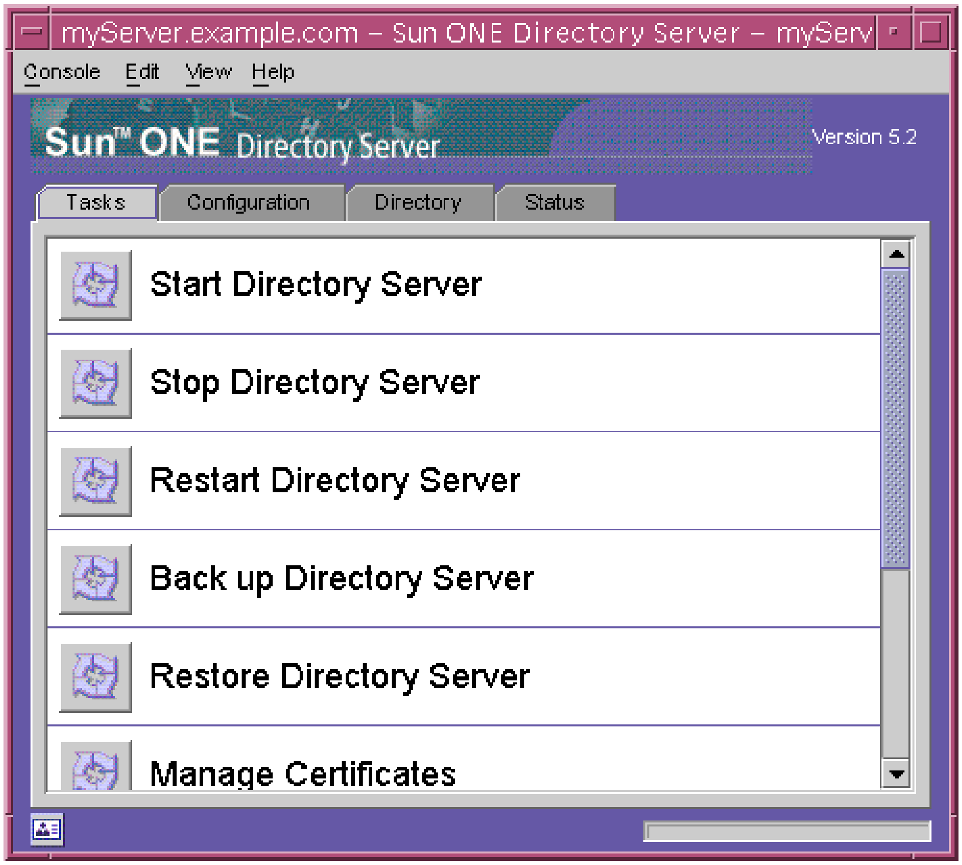
   Sun ONE Server Console is displayed.

5. Navigate through the tree in the left-hand pane to find the machine hosting your Directory Server and click on its name or icon to display its general properties.

6. Double-click the name of your Directory Server in the tree or click the Open button to display Directory Server Console for managing this Directory Server instance.

   Directory Server Console for example.com is shown in Figure 3-1 on page 53.

Figure 3-1 Directory Server Console



Managing Entries

It is unlikely that you will use the console to add, delete or modify bulk entries. These processes are generally performed using the command-line utilities. To understand how entries are stored in the directory, however, it is useful to use the console to add a single entry.

Adding an Entry

As a simple exercise, add a user to the default database userRoot.

1. In Directory Server Console, select the Directory tab.
2. Expand the `example` node. This is the default suffix that you created at installation.

   The `example` node contains three subnodes: Groups, People and Special Users. Select the People node.
3. From the Object menu, select New > User. You can also right-click on the People node and select New > User.
4. Enter the First Name and Last Name of the new user. For the purposes of this example, use the name Barbara Jensen.
5. Click OK.
6. Select View > Refresh and click on the People node to check that your new user has been added.

Finding Entries

Use the Directory tab of Directory Server Console to browse the contents of the directory tree and search for specific entries in the directory.

1. On Directory Server Console, select the Directory tab.

   Depending on the DN you used to authenticate to the directory, this tab displays the contents of the directory that you have access permissions to view. You can browse through the contents of the tree or right-click an entry and select Search from the pop-up menu.

2. The Search dialog provides a simple interface for finding entries in the directory. This dialog performs a search from the node in the directory that was selected when the dialog was invoked. Search from the highest level of the directory for a wider search, or from a lower subtree for a quicker search.

   The Advanced Search allows you to refine your search to certain attributes and their values. The Filtered Search is available if you want to search with your own LDAP filter string.

   See the online help for further information on using this feature.

Understanding the Schema

The schema defines the size, range and format of entries that are stored in the directory. When you add or change an entry in the directory, the schema determines whether the object class is valid. It also determines the required attributes and the optional attributes for the object class.

Looking at the Schema in the Console

1. In the Directory Server Console, select the Configuration tab.
2. Click on the Schema node and select the Object Classes tab.
3. In the Standard Object Classes pane, scroll down and select the `person` Object Class.

   Note that the required attributes are `cn` (common name), `sn` (surname) and `objectClass`. This means that when you add an entry of type `person`, you must add the `cn`, `sn` and `objectClass` attributes.

NOTE The console prevents you from adding entries that violate the schema.

Working with Groups and Roles

As discussed in the previous chapter, groups provide associations between entries. Static groups explicitly name their member entries. Dynamic groups specify a filter and all entries that match are members of the group.

Roles are designed to be more efficient and easier to use for applications. Roles are defined and administered like groups, but in addition, member entries also have a generated attribute that indicates the roles in which they participate.

In the following exercises, we will add the person created previously to one of the default groups in the `example` database. We will also create a new managed role.

Adding a Member to a Static Group

1. In Directory Server Console, select the Directory tab.
2. Expand the `example` node. This is the default suffix that you created at installation.

   The `example` node contains three subnodes: Groups, People and Special Users.
3. Select the Groups node to display the four default groups that appear in the `example` database.
4. Double-click the HR Managers group.
5. Click the Members node to display the members of the group. There should be no members defined for the group.
6. On the Static Group tab click Add to add a new member.
7. Select Users from the Search dropdown list and click Search.

   The only user entry we have added to the directory (Barbara Jensen) is displayed.
8. Click OK.

   Barabara Jensen is now added as a member of the static group HR Managers.
9. Click OK to return to the Directory tab.

Creating a Managed Role

Managed roles allow you to create an explicit enumerated list of members (much like a static group). Managed roles are added to entries by adding the `nsRoleDN` attribute to the entry.

To create and add members to a managed role:

1. In Directory Server Console, select the Directory tab.
2. Browse the directory tree and select the parent entry for your new role.

   For the purposes of simplicity, we will select the People entry in this exercise, although this may not be realistic in an actual deployment.
3. From the Object menu, select New > Role. You can also right click the entry and select New > Role.

   The Create New Role dialog box is displayed.
4. Click General in the left pane. Type a name for your new role in the "Role Name" field. For the purposes of this exercise, use `myCorpAdminRole` as the role name.

   The role name is required.
5. Enter a description of the new role in the "Description" field. For the purposes of this exercise, enter “corporate administrators” here.
6. Click Members in the left pane.
7. In the right pane, select Managed Role. Click Add to add new entries to the list of members.

   The standard "Search users and groups" dialog box is displayed.
8. Select Users from the Search dropdown list, then click Search. Select the only user entry currently in the directory (Barbara Jensen) and click OK.
9. Click OK again to dismiss the Create New Role window.

   Barbara Jensen is now a member of the role `myCorpAdminRole`. The new role appears in the directory with the icon for a managed role.

Checking Role Membership

Because managed roles add the `nsRoleDN` attribute to an entry, it is easy to see the managed roles to which an entry user belongs. To check Barbara Jensen's membership of `myCorpAdminRole`:

1. In Directory Server Console, select the Directory tab.
2. Expand the `example` node and click the People node.
3. Select Barabara Jensen's entry in the right pane.
4. Select Edit With Generic Editor from the Object menu.

   The Generic Editor window is displayed.
5. Note that Barbara Jensen's entry now has an `nsRoleDN` attribute whose value is `cn=myCorpAdminRole,ou=People,dc=example,dc=COM`.

Working With Class of Service (CoS)

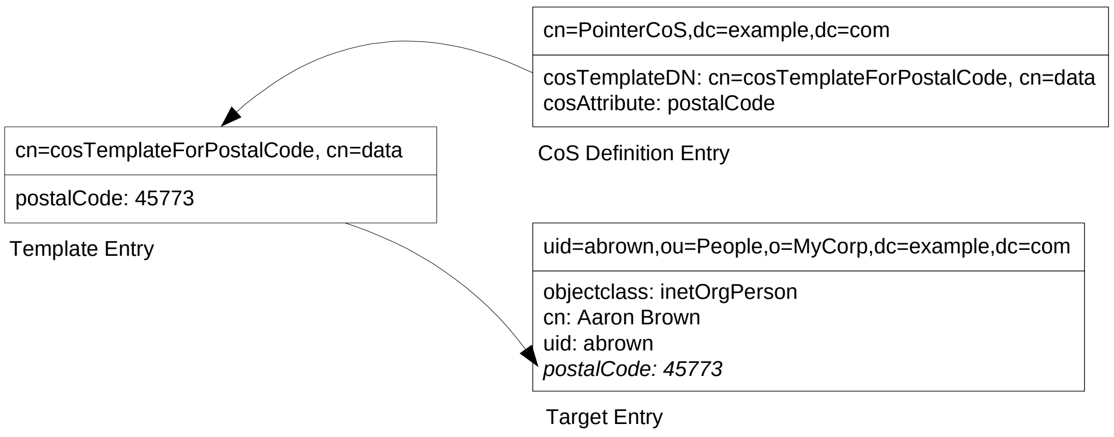
The Class of Service (CoS) mechanism allows you to create virtual attributes not stored in the entries. Instead, they are generated by the CoS mechanism as the entry is sent to the client application. CoS simplifies entry management and reduces storage requirements.

NOTE CoS functionality is subject to certain restrictions. See the *Sun ONE Directory Server Administration Guide* and the *Sun ONE Directory Server Deployment Guide* for a complete understanding of CoS.

In brief, a CoS defines a virtual attribute and its value for all of its target entries, any entry within the scope of the CoS. Each CoS is comprised of a CoS Definition Entry and a CoS Template Entry. Creating a new CoS implies creating a definition entry and a template entry.

In the following exercise, we will create a CoS that defines a common postal code for all of the entries stored under `dc=example,dc=com`. The following diagram shows the entries that would be affected by this example.

Figure 3-2 Pointer CoS Definition, Template and Target Entries


cn=PointerCoS,dc=example,dc=com
cosTemplateDN: cn=cosTemplateForPostalCode, cn=data
cosAttribute: postalCode
CoS Definition Entry
cn=cosTemplateForPostalCode, cn=data
postalCode: 45773
Template Entry
uid=abrown,ou=People,o=MyCorp,dc=example,dc=com
objectclass: inetOrgPerson
cn: Aaron Brown
uid: abrown
postalCode: 45773
Target Entry

Creating a CoS Template Entry

1. In the Directory Server Console, select the Directory tab.
2. Browse the directory tree and select the parent entry under which you wish to store the template entry. For the purposes of this exercise, select People.
3. From the Object menu (or the right-click context menu), select New > Other, and then select `costemplate` from the list in the New Object dialog.
4. Click OK.

   The Generic Editor dialog opens with default values for certain attributes in the new template.
5. Edit the new template object as follows:
   a. Click any value next to the Object class attribute and click Add Value to add the `ldapsubentry` and `extensibleobject` values to the objectclass attribute.
   b. Click Add Attribute to add the `cn` attribute (the attribute appears as `Full Name` in the list of attributes). Click in the text area to the right of the attribute and enter a value that will identify the template. For this example, we will use `cosTemplateForPostalCode`.

c. Change the naming attribute to the new cn attribute. To do this, click the Change button, clear the In Naming Attribute checkbox for the cospriority attribute, and select the In Naming Attribute checkbox for the cn attribute.

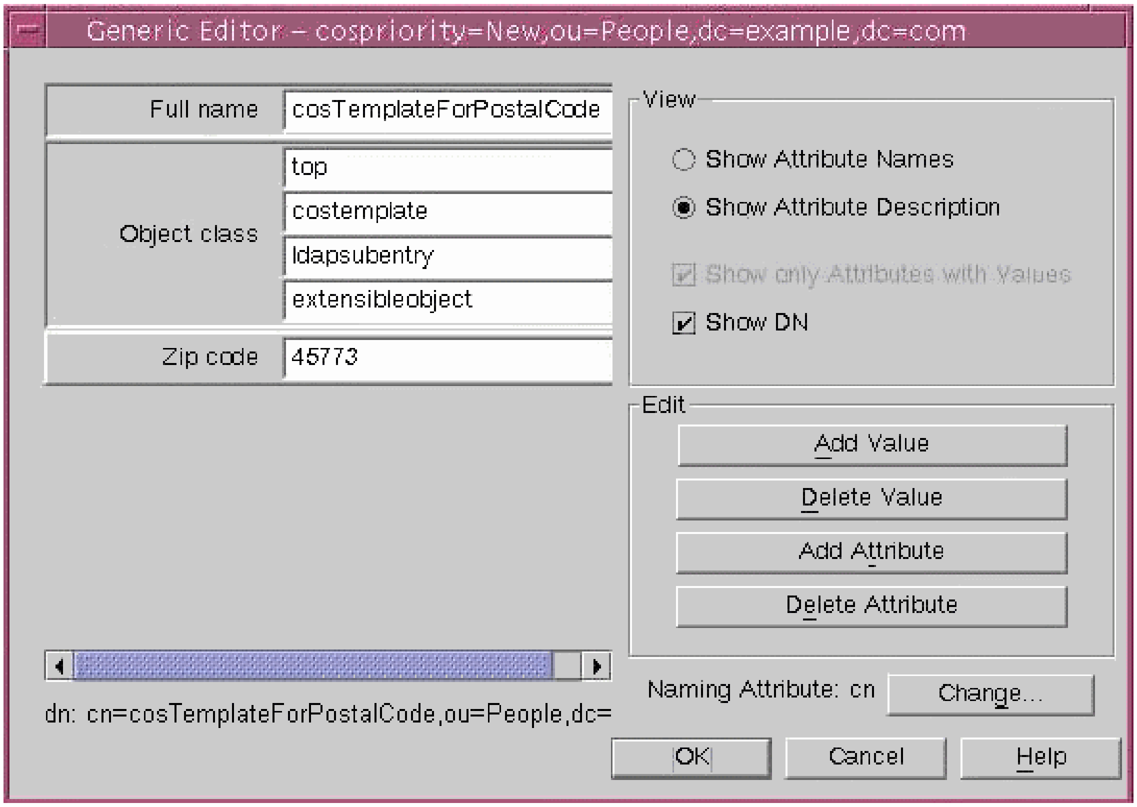
d. Click OK.

e. Delete the cosPriority attribute. It is not required for the purposes of this example. To do this, select the cosPriority attribute and click Delete Attribute.

f. Add the attribute and its value to generate on target entries by the CoS mechanism. Since we are creating a CoS that will specify the postal code of entries, add the postalcode attribute and enter the value of the postal code. We will use 45773.

The Generic Editor dialog appears, as shown in Figure 3-3:

Figure 3-3 Generic Editor Dialog


Generic Editor – cospriority=New,ou=People,dc=example,dc=com

Full name	cosTemplateForPostalCode
Object class	top
	costemplate
	ldapsubentry
	extensibleobject
Zip code	45773

View
( ) Show Attribute Names
(●) Show Attribute Description
[✓] Show only Attributes with Values
[✓] Show DN

dn: cn=cosTemplateForPostalCode,ou=People,dc=

Naming Attribute: cn

6. Click OK in the Generic Editor dialog to create the template entry.

   The new CoS appears in the right pane.

7. To define a pointer CoS for this template, select the new template entry in directory tree and select Copy DN from the Edit menu.

Creating a CoS Definition Entry

1. Browse the directory tree and select the parent entry under which you wish the new Class of Service to take effect. For the purposes of this exercise, select People.

2. From the Object menu or the right-click context menu, select New > Class of Service.

   The Create New Class of Service dialog is displayed.

3. Select General in the left pane. In the right pane, enter the name of your new Class of Service in the "Class Name" field. The name will appear in the `cn` naming attribute for the CoS definition entry. Since we are creating a CoS that will generate the postal code, we will call this CoS `cosGeneratePostalCode`. Enter a description in the "Description" field.

4. Click Attributes in the left pane. The right pane displays the list of attributes that will be generated by the CoS mechanism on the target entries.

5. Click Add to locate the `postalcode` attribute and add it to the list.

6. Once you have added an attribute to the list, the "Class of Service Behavior" column contains a drop-down list. Click in this cell and select “Overrides target entry attribute”. The value of the attribute generated by the CoS will now override any value for that attribute in the target entry.

7. Click Template in the left pane. In the right pane, select how the template entry is identified and then fill in the corresponding fields. This will determine the type of CoS you wish to define.

   For this exercise, select “By its DN”. This will define a pointer CoS. Type Ctrl-V to paste the DN that you copied after creating the template entry.

8. Click OK to create the CoS definition entry.

The CoS template and the CoS definition are now displayed in the right pane when you select the People node.

Working With ACIs

Access Control is one of the primary methods of making a directory secure. Using access control, you can control access to the entire directory, a subtree of the directory, specific entries in the directory (including entries defining configuration tasks), or a specific set of entry attributes. You can set permissions for a specific user, all users belonging to a specific group or role, or all users of the directory. Finally, you can define access for a specific location such as an IP address or a DNS name.

Creating a New ACI

In this exercise, we will create an ACI that gives full access rights to Barabara Jensen, the user we created previously. Note that this exercise does not explain the steps in detail. Refer to the *Sun ONE Directory Server Administration Guide* for a complete explanation of this process and the theory behind each step.

1. In Directory Server Console, select the Directory tab.
2. Select the People entry and select Set Access Permissions from the Object menu.
3. In the Manage Access Control dialog, select Allow self entry modification in the left pane and click New to create a new ACI.
4. Type a name for the ACI in the ACI name text box.

   The name can be any string you want to use to identify the ACI. For this exercise, use the name **`Full rights for Barabara`**.
5. On the Users/Groups tab, select All users and click Remove. Click the Add button to the one user to whom this ACI will apply.
6. In the Add Users and Groups window:

   a. Select the Users and Groups search area from the drop-down list, enter **`BJensen`** in the Search field, and click the Search button.

   b. Highlight Barabara Jensen’s entry in the search result list, and click the Add button to add it to the list of entries which have access permission.

   c. Click OK to dismiss the Add Users and Groups window.

   Barabara Jensen’s entry is now listed on the Users/Groups tab in the Edit ACI dialog.

7. Select the Rights tab, and click the Check All button to specify that this user should be given full rights. Note that, as a security precaution, the proxy checkbox is not selected when you click Check All. For the purposes of this example, it is not necessary to add proxy authorization to the ACI.
8. Click the Targets tab, then click This Entry to display the node targeted by the ACI.
9. Click OK to save the new ACI.

The new ACI is listed in the ACI Manager window.

Setting Up Replication

As described in the previous chapter, replication is the mechanism that automatically copies directory data from one Directory Server instance to another. In this section, we will set up a basic multi-master replication scenario.

To set up replication in our sample installation, we will create a second Directory Server instance on the same host. Note that this setup is for the purposes of demonstrating the replication functionality. In a standard deployment scenario it is unlikely that you would set up replication between two Directory Server instances on the same server.

Creating a New Server Instance

1. In the Sun ONE Server Console, select the Server Group node.
2. Select Create Instance Of > Sun ONE Directory Server from the Object menu.
3. In the Create New Instance window, enter the Server Identifier and Network Port for the new server instance. For this example, use **`myServer2`** for the Server Identifier and **`5202`** for the Network Port.
4. For the remaining fields, use the same information that you used when installing Sun ONE Directory Server.
5. Click OK to continue.

   The Status window indicates that the new server instance has been created and that the server has been started.

6. Click OK to close the status window.

   The two Directory Server instances should now be visible under the Server Group node. For the purposes of this example, we will refer to the first server instance as *myServer* and the second as *myServer2*.

Creating a New Suffix

To demonstrate the replication functionality, create a new suffix on each server instance.

1. In the Directory Server Console of *myServer*, select the Configuration tab.
2. Select the Data node and select New Suffix from the Object menu.
3. In the New Suffix window, enter the following as the suffix DN: **`dc=repl,dc=example,dc=com`**.
4. Click OK.

   A status window is displayed, indicating the progress of the suffix creation.
5. Expand the Data node and check that the new suffix has been created.
6. Repeat this process for the *myServer2* server instance (using the same suffix DN).

Creating the Data Object

The new suffix on each server instance does not yet exist as an *object* in the directory. To create a new object in the directory, you must log in as the Directory Manager:

1. In the Directory Server Console of *myServer*, select Log in as New User from the Console menu.
2. Enter the Distinguished Name (**`cn=directory manager`**) and the password (**`password`**) that you set up when you installed Directory Server.

To set the suffix up as a data object:

1. Select the Directory tab.
2. Select the server instance node (myServer.example.com).
3. Select New Root Object from the Object menu and select the suffix you created in the previous procedure (dc=repl,dc=example,dc=com).

4. In the New Object window, select domain and click OK.

   The new object is displayed in the Generic Editor.

5. Click OK to continue.

   Note that the object (repl) appears as a new node in the left pane. Repeat this process for the *myServer2* server instance.

Enabling Replication

When you first create a new suffix, replication is disabled by default. In the Directory Server console of *myServer*, expand the new suffix node (under the Data node). You will notice that the Replication node is marked Disabled. To enable replication:

1. Select the Replication node below the suffix name.
2. Click Enable Replication in the right-hand pane.
3. As we are setting up a multi-master replication scenario (both the server instances will be masters), select Master Replica and click Next.
4. Enter an ID for the replica. The replica ID is a 16 bit integer between 1 and 65534 and must be unique for all master replicas. Consumer replicas always have the same replica ID (65535.) For this example, enter **`1`** and click Next.
5. Specify the location of the change log database. For this example, accept the default (`/`*ServerRoot*`/slapd-myServer/changelogdb`) and click Next.
6. Enter a password for the replication manager. The replication manager is the bind DN that will be used to bind to the other server instance during replication. For this example, enter the password **`replpword`**.
7. Enter the password again to confirm it and click Next.

   A status box is displayed, indicating that replication is being enabled.

8. Click Close.

   Note that the Replication node no longer states that replication is disabled.

9. Repeat this process for the *myServer2* server instance, using a replica ID of **`2`**.

Setting up the Replication Agreement

Once you have set up the master replica, you must specify the replicas to which that master will send updates. This specification is called the Replication Agreement.

1. On the Configuration tab of *myServer*, expand the new suffix node (under the Data node) and select the Replication node.
2. Click New on the right-hand pane.
3. In the Replication Agreement window, select `myServer2.example.com:5202` as the consumer replica to which updates will be sent.
4. In the Authentication frame, accept the default DN (`cn=replication manager,cn=replication,cn=config`) and enter the replication password you chose previously (`replpword`).
5. Click OK.

   A confirmation message is displayed, asking whether you want to check the authentication details you have entered.
6. Click Yes to check that the master server can indeed contact the consumer.

   An informational message is displayed, indicating that the master server can contact the consumer server.
7. Click OK.

   The consumer replica has now been set up.
8. Repeat this process for the *myServer2* server instance, selecting `myServer.example.com:5201` as the consumer replica.

Initializing the Consumer

Once the Replication Agreement has been set up, you can initialize the consumer replica. Initializing the consumer copies the data that is on the master replica to the consumer replica. Once this has been done, only changes made to the master replica will be copied to the consumer replica.

To initialize the consumer:

1. On the Configuration tab of *myServer*, expand the new suffix node (under the Data node) and select the Replication node.

2. Select the consumer in the right hand pane (`myServer2.example.com:5202`) and click Action.
3. Select Initialize remote replica.
4. Click Yes to confirm that you want the existing content on the remote replica to be cleared.

   The consumer replica has now been initialized.

Testing the Replication

To test whether the replication setup has been successful:

1. Add an entry to the *repl* database on *myServer*. For information on how to do this, see “Adding an Entry,” on page 54.
2. Check the same database on *myServer2* and notice that the change you made has been replicated to the second server instance.

   Now add an entry to the database on *myServer2*. Because this is a multi-master replication scenario, changes are accepted on both servers and are replicated to both consumers.

Chapter 4

A Quick Look at Directory Server Command-Line Utilities

This chapter provides practical examples of the features described in Chapter 2, "Introduction to Sun ONE Directory Server", using the command-line utilities. It also contains basic information on some of the files installed with Sun ONE Directory Server that will help you understand where default directory data is located and how it can be accessed and modified.

This chapter covers the following topics:

- Examining the Configuration
- Examining the Data
- Adding, Changing and Deleting Entries
- Working With the Schema
- Working With Groups and Roles
- Working With Class of Service (CoS)
- Working With ACIs
- Examining the Log Information
- Searching an Internationalized Directory

NOTE This chapter assumes that you have successfully installed Sun ONE Directory Server. For more information on how to do this, see "Installing and Configuring Directory Server," on page 50.

Examining the Configuration

Directory Server configuration information is held in the file `dse.ldif`, located in the *ServerRoot*/`slapd-`*serverID*/`config` directory. In our sample installation performed in the previous chapter, the configuration file is `slapd-myServer/config/dse.ldif`. Directory Server requires this file at startup.

Locate the configuration file in your installation and open the file in a text editor. (On Windows systems, use an editor other than Notepad.)

The following is an extract of the `dse.ldif` file from the sample installation performed in the previous chapter.

Code Example 4-1 Extract From Sample `dse.ldif` File

```
dn:
objectClass: top
aci: (targetattr != "aci")(version 3.0; aci "rootdse anon read
access"; allow(read,search,compare) userdn="ldap:///anyone";)
creatorsName: cn=server,cn=plugins,cn=config
modifiersName: cn=server,cn=plugins,cn=config
createTimestamp: 20020428110553Z
modifyTimestamp: 20020428110553Z

dn: cn=config
cn: config
objectClass: top
objectClass: extensibleObject
objectClass: nsslapdConfig
nsslapd-accesslog-logging-enabled: on
nsslapd-accesslog: ServerRoot/slapd-myServer/logs/access
nsslapd-accesslog-maxlogsperdir: 10
nsslapd-accesslog-maxlogsize: 100
nsslapd-accesslog-logrotationtime: 1
nsslapd-accesslog-logrotationtimeunit: day
nsslapd-enquote-sup-oc: off
nsslapd-localhost: myServer.example.COM
```

Examine the `dse.ldif` file from your own installation. Note the default configuration attributes and their values.

NOTE To change attribute values, do not modify the `dse.ldif` file directly. Instead, use the `ldapmodify` client utility, described in this section, then restart Directory Server to force it to reread the `dse.ldif` file.

Examining the Data

As described in Chapter 2, Sun ONE Directory Server is installed with two default databases:

- *userRoot*, the default database you selected at installation.
- *NetscapeRoot*, a database that contains the configuration information of other Sun ONE servers, such as Sun ONE Administration Server.

Both databases are located in the `slapd-`*serverID*`/db` directory. In our installation, these files are located in `slapd-myServer/db`. Locate the database directory in your installation and check that the two default databases are there.

LDIF File Format

Directory Server uses the LDAP Data Interchange Format (LDIF) to describe a directory and directory entries in text format. LDIF is commonly used to build the initial directory database or to add large numbers of entries to the directory all at once. In addition, LDIF is also used to describe changes to directory entries. For this reason, most of Directory Server's command-line utilities rely on LDIF for either input or output. For more information on the LDIF file format, see LDIF File Format in the *Sun ONE Directory Server Reference Manual*.

LDIF consists of one or more directory entries separated by a blank line. Each LDIF entry consists of an optional entry ID, a required distinguished name, one or more object classes, and multiple attribute definitions.

The basic form of a directory entry represented in LDIF is as follows:

```
dn: distinguished_name
objectClass: object_class
objectClass: object_class
...
attribute_type[;subtype]:attribute_value
attribute_type[;subtype]:attribute_value
...
```

NOTE Do not leave white space unintentionally at the end of an LDIF attribute value string. When Directory Server reads an attribute value ending in white space, it base 64 encodes the value.

Sample ldif File

The following is a small sample ldif file, `data.ldif`. This file will be used to create a database in the next section. Note that this sample file contains the following:

- One organization (MyCorp)
- One group (Administrators)
- Three people (Aaron Brown, Brian Crane and Charlene Daniels)

```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example

dn: o=MyCorp,dc=example,dc=com
objectclass: top
objectclass: organization
o: MyCorp

dn: ou=Groups,o=MyCorp,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: Groups

dn: cn=Administrators,ou=Groups,o=MyCorp,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Administrators
uniqueMember: uid=abrown,ou=People,o=MyCorp,dc=example,dc=com
uniqueMember: uid=bcrane,ou=People,o=MyCorp,dc=example,dc=com

dn: ou=People,o=MyCorp,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: People

dn: uid=abrown,ou=People,o=MyCorp,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: abrown
givenName: Aaron
sn: Brown
cn: Aaron Brown
mail: abrown@mycorp.com
userPassword: abrown
```

```
facsimiletelephonenumber: 666

dn: uid=bcrane,ou=People,o=MyCorp,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: bcrane
givenName: Brian
sn: Crane
cn: Brian Crane
mail: bcrane@mycorp.com
userPassword: bcrane
secretary: uid=abrown,ou=People,o=MyCorp,dc=example,dc=com

dn: uid=cdaniels,ou=People,o=MyCorp,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: cdaniels
givenName: Charlene
sn: Daniels
cn: Charlene Daniels
mail: cdaniels@mycorp.com
userPassword: cdaniels
secretary: uid=abrown,ou=People,o=MyCorp,dc=example,dc=com
```

Creating a New Suffix

Before you can import data into a new database, you must set up the database configuration by creating a new suffix. In the following example, we will create a new suffix called *example*. Creating a new suffix at the command-line involves using the `ldapmodify` command to add an LDIF entry to the directory.

To create a new backend and suffix for the sample database:

1. At the command-line type:

```
ldapmodify -a -p 5201 -D "cn=directory manager" -w password
dn: cn="dc=example,dc-com",cn=mapping tree,cn=config
objectClass: top
objectClass: extensibleObject
objectClass: nsMappingTree
```

```
cn: example
nsslapd-state: backend
nsslapd-backend: example
```

Remember that `5201` was the port specified at installation and `password` was the password specified for the directory manager. If you used alternative values at installation, replace these with the values you used.

After the last line, hit Return and Ctrl-D (on UNIX platforms) or Ctrl-Z (on Windows platforms) to indicate the end the file.

2. At the command-line type:

```
ldapmodify -a -p 5201 -D "cn=directory manager" -w password
dn: cn=example,cn=ldbm database,cn=plugins,cn=config
objectClass: top
objectClass: extensibleObject
objectClass: nsBackendInstance
cn: example
nsslapd-suffix: dc=example,dc=com
```

After the last line, hit Return and Ctrl-D (on UNIX platforms) or Ctrl-Z (on Windows platforms) to indicate the end the file.

NOTE The `ldapmodify` command is explained in detail later in this chapter. For now, it is sufficient to understand that the above command adds a suffix entry to the directory.

Populating a New Database Using ldif2db

Data that is contained in an LDIF file can be imported into the directory as a new database. In this exercise, we will use the `ldif2db` utility to load a sample database. For a complete description of `ldif2db`, refer to Chapter 4, “Populating Directory Databases” in the *Sun ONE Directory Server Administration Guide*.

The `ldif2db` utility is located in the `slapd-`*serverID*`/` directory. In our installation it is located in `slapd-myServer/`.

1. Create the `data.ldif` file from the sample file above. (In a text editor other than Notepad, copy the sample file and save it as /*path*/`data.ldif,` where *path* can be any location to which you have read and write access).

2. Change to *ServerRoot*`/slapd-myServer/`.

3. Before running the `ldif2db` conversion, you must stop the server, as follows:

   ```
   % stop-slapd
   ```

4. When the server is stopped, type the following to import the new data:

   ```
   % ldif2db -s dc=example,dc=com -i /path/data.ldif
   ```

5. The progress of the import is displayed. The message "Import complete" is displayed when the conversion is complete. For the sample database above, you should see a message indicating that 8 entries were processed.

6. Restart the server as follows:

   ```
   % start-slapd
   ```

Searching the Directory

You can locate entries in a directory using any LDAP client. Most clients provide some form of search interface that enables you to search the directory and retrieve entry information.

NOTE The access control that has been set in your directory determines the results of your searches. Common users typically do not "see" much of the directory, and directory administrators have full access to all data, including configuration.

Searching the Directory With ldapsearch

You can use the `ldapsearch` command-line utility to locate and retrieve directory entries. Note that the `ldapsearch` utility described in this section is not the utility provided with the Solaris platform, but is part of the Sun ONE Directory Server Resource Kit. For more information on this utility, refer to the *Sun ONE Directory Server Resource Kit Tools Reference*.

This utility opens a connection to the server using the specified distinguished name and password, and locates entries based on a specified search filter. Search scopes can include a single entry, an entry's immediate subentries, or an entire tree or subtree.

Search results are returned in LDIF format.

ldapsearch Command-Line Format

When you use `ldapsearch`, you must enter the command using the following format:

`ldapsearch` [*optional_options*] [*optional_search_filter*] [*optional_list_of_attributes*]

where

- *optional_options* represents a series of command-line options. These must be specified before the search filter, if any.
- *optional_search_filter* represents an LDAP search filter as described in "LDAP Search Filters," on page 81. Do not specify a search filter if you are supplying search filters in a file using the `-f` option.
- *optional_list_of_attributes* represents a list of attributes separated by a space. Specifying a list of attributes reduces the number of attributes returned in the search results. This list of attributes must appear after the search filter. For an example, see "Displaying Subsets of Attributes," on page 80. If you do not specify a list of attributes, the search returns values for all attributes permitted by the access control set in the directory (with the exception of operational attributes).

NOTE If you want operational attributes returned as a result of a search operation, you must explicitly specify them in the search command. To retrieve regular attributes in addition to explicitly specified operational attributes, use an asterisk (*) in the list of attributes in the `ldapsearch` command.

Using Special Characters

When using the `ldapsearch` command-line utility, you may need to specify values that contain characters that have special meaning to the command-line interpreter (such as space [ ], asterisk [*], backslash [\], and so forth). When you specify special characters, enclose the value in quotation marks (""). For example:

```
-D "cn=Charlene Daniels,ou=People,dc=example,dc=com"
```

Depending on your command-line interpreter, use either single or double quotation marks for this purpose. Refer to your shell documentation for more information.

Commonly Used ldapsearch options

The following lists the most commonly used `ldapsearch` command-line options. If you specify a value that contains a space [ ], the value should be surrounded by double quotation marks, for example, `-b "ou=groups, dc=example,dc=com"`.

`-b`	Specifies the starting point for the search. The value specified here must be a distinguished name that currently exists in the database. This option is optional if the `LDAP_BASEDN` environment variable has been set to a base DN. The value specified in this option should be provided in double quotation marks. For example: `-b "cn=Charlene Daniels, ou=People, dc=example,dc=com"`
`-D`	Specifies the distinguished name with which to authenticate to the server. This option is optional if anonymous access is supported by your server. If specified, this value must be a DN recognized by the Directory Server, and it must also have the authority to search for the entries. For example: `-D "uid=cdaniels, dc=example,dc=com"`
`-h`	Specifies the hostname or IP address of the machine on which the Directory Server is installed. If you do not specify a host, `ldapsearch` uses the localhost. For example, `-h myServer`.
`-l`	Specifies the maximum number of seconds to wait for a search request to complete. Regardless of the value specified here, `ldapsearch` will never wait longer than is allowed by the server's `nsslapd-timelimit` attribute (except in the case of a persistent search.) For more information on persistent searches, see Chapter 3 "ldapsearch" in the *Sun ONE Directory Server Resource Kit Tools Reference*. For example, `-l 300`. The default value for the `nsslapd-timelimit` attribute is 3,600 seconds (1 hour.)
`-p`	Specifies the TCP port number that the Directory Server uses. For example, `-p 5201`. The default is 389. If `-Z` is used, the default is 636.

`-s`	Specifies the scope of the search. The scope can be one of the following: • `base`—Search only the entry specified in the `-b` option or defined by the `LDAP_BASEDN` environment variable. • `one`—Search only the immediate children of the entry specified in the `-b` option. Only the children are searched; the actual entry specified in the `-b` option is not searched. • `sub`—Search the entry specified in the `-b` option and all of its descendants. That is, perform a subtree search starting at the point identified in the `-b` option. This is the default.
`-w`	Specifies the password associated with the distinguished name that is specified in the `-D` option. If you do not specify this option, anonymous access is used. For example, `-w diner892`.
`-x`	Specifies that the search results are sorted on the server rather than on the client. This is useful if you want to sort according to a matching rule, as with an international search. In general, it is faster to sort on the server rather than on the client, although server-side sorting uses server resources.
`-z`	Specifies the maximum number of entries to return in response to a search request. For example, `-z 1000`. Normally, regardless of the value specified here, `ldapsearch` never returns more entries than the number allowed by the server's `nsslapd-sizelimit` attribute. However, you can override this limitation by binding as the root DN when using this command-line argument. When you bind as the root DN, this option defaults to zero (0). The default value for the `nsslapd-sizelimit` attribute is 2,000 entries.

For detailed information on all `ldapsearch` utility options, refer to the *Sun ONE Directory Server Resource Kit Tools Reference*.

ldapsearch Examples

In the next set of examples, the following assumptions are made:

- You want to perform a search of all entries in the directory.
- The server is located on hostname **`myServer`**.
- The server uses port number **`5201`**.
- You are binding to the directory as Directory Manager, with a password of **`password`**.
- SSL is enabled for the server on port **`636`** (the default SSL port number).
- The suffix under which all data is stored is **`dc=example,dc=com`**.

Returning All Entries

Given the previous information, the following call will return all entries in the directory:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -b
"dc=example,dc=com" -s sub "objectclass=*"
```

"`objectclass=*`" is a search filter that matches any entry in the directory.

Specifying Search Filters on the Command Line

You can specify a search filter directly on the command line. If you do this, be sure to enclose your filter in quotation marks ("filter"). Also, do not specify the `-f` option. For example:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -b
"dc=example,dc=com" "cn=Charlene Daniels"
```

Searching the Root DSE Entry

The root DSE, is a special entry that contains a list of all the suffixes supported by the local directory server. You can search this entry by supplying a search base of "". You must also specify a search scope of `base` and a filter of `"objectclass=*"`. For example:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -b "" -s base
"objectclass=*"
```

Searching the Schema Entry

Sun ONE Directory Server stores all directory server schema in the special `cn=schema` entry. This entry contains information on every object class and attribute defined for your directory server.

You can examine the contents of this entry as follows:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -b
"cn=schema" -s base "objectclass=*"
```

Using LDAP_BASEDN

To make searching easier, you can set your search base using the `LDAP_BASEDN` environment variable. Doing this allows you to skip specifying the search base with the `-b` option (for information on how to set environment variables, see the documentation for your operating system).

Typically, you set `LDAP_BASEDN` to your directory's suffix value. Since your directory suffix is equal to the root, or topmost, entry in your directory, this causes all searches to begin from your directory's root entry.

For example, if you have set `LDAP_BASEDN` to `dc=example,dc=com`, you can search for `cn=Charlene Daniels` in your directory using the following command-line call:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password "cn=Charlene
Daniels"
```

In this example, the default scope of `sub` is used because the `-s` option was not used to specify the scope.

Displaying Subsets of Attributes

The `ldapsearch` command returns all search results in LDIF format. By default, `ldapsearch` returns the entry's distinguished name and all of the attributes that you are allowed to read. You can set up the directory access control such that you are allowed to read only a subset of the attributes on any given directory entry.) Only operational attributes are not returned. If you want operational attributes returned as a result of a search operation, you must explicitly specify them in the search command. For more information on operational attributes, refer to Chapter 12, "Operational Attributes" in the *Sun ONE Directory Server Reference Manual*.

Suppose you do not want to see all of the attributes returned in the search results. You can limit the returned attributes to just a few specific attributes by specifying the ones you want on the command line immediately after the search filter. For example, to show the `cn` and `sn` attributes for every entry in the directory, use the following command:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password
"objectclass=*" sn cn
```

This example assumes you set your search base with `LDAP_BASEDN`.

LDAP Search Filters

Search filters select the entries to be returned for a search operation. They are most commonly used with the `ldapsearch` command-line utility. When you use `ldapsearch`, you can place multiple search filters in a file, with each filter on a separate line in the file, or you can specify a search filter directly on the command line.

For example, the following filter specifies a search for the common name Charlene Daniels:

```
cn=Charlene Daniels
```

This search filter returns all entries that contain the common name Charlene Daniels. Searches for common name values are not case sensitive.

When the common name attribute has values associated with a language tag, all of the values are returned. Thus, the following two attribute values both match this filter:

```
cn: Monique Le Chat
cn;lang-fr: Monique Le Chat
```

Search Filter Syntax

The basic syntax of a search filter is:

attribute operator value

For example:

```
buildingname>=alpha
```

In this example, `buildingname` is the attribute, >= is the operator, and **`alpha`** is the value. You can also define filters that use different attributes combined together with Boolean operators.

Search filters are described in detail in the following sections:

- Using Attributes in Search Filters
- Using Operators in Search Filters
- Using Compound Search Filters
- Search Filter Examples

Using Attributes in Search Filters

When searching for an entry, you can specify attributes associated with that type of entry. For example, when you search for people entries, you can use the `cn` attribute to search for people with a specific common name.

Examples of attributes that people entries might include:

- `cn` (the person's common name)
- `sn` (the person's surname, or last name, or family name)
- `telephoneNumber` (the person's telephone number)
- `buildingName` (the name of the building in which the person resides)
- `l` (the locality in which you can find the person)

For a listing of the attributes associated with types of entries, see the *Sun ONE Directory Server Reference Manual*.

Using Operators in Search Filters

The operators that you can use in search filters are listed in Table 4-1:

Table 4-1 Search Filter Operators

Search type	Operator	Description
Equality	=	Returns entries containing attribute values that exactly match the specified value. For example, `cn=Bob Johnson`

Table 4-1 Search Filter Operators *(Continued)*

Search type	Operator	Description

Substring	*=string* string*	Returns entries containing attributes containing the specified substring. For example,
		`cn=Bob*`
		`cn=*Johnson`
		`cn=*John*`
		`cn=B*John`
		(The asterisk (*) indicates zero (0) or more characters.)

Greater than or equal to	>=	Returns entries containing attributes that are greater than or equal to the specified value. For example, `buildingname >= alpha`
Less than or equal to	<=	Returns entries containing attributes that are less than or equal to the specified value. For example, `buildingname <= alpha`

Presence	=*	Returns entries containing one or more values for the specified attribute. For example,
		`cn=*`
		`telephonenumber=*`
		`manager=*`

Approximate	~=	Returns entries containing the specified attribute with a value that is approximately equal to the value specified in the search filter. For example,
		`cn~=suret`
		`l~=san fransico`
		could return
		`cn=sarette`
		`l=san francisco`

NOTE In addition to these search filters, you can specify special filters to work with a preferred language collation order. For information on how to search a directory with international character sets, see "Searching an Internationalized Directory," on page 98.

Using Compound Search Filters

Multiple search filter components can be combined using Boolean operators expressed in prefix notation as follows:

(*Boolean-operator*(*filter*)(*filter*)(*filter*)...)

where *Boolean-operator* is any one of the Boolean operators listed in Table 4-2.

Boolean operators can be combined and nested together to form complex expressions, such as:

(*Boolean-operator*(*filter*)(*Boolean-operator*(*filter*)(*filter*)))

The Boolean operators available for use with search filters include the following:

Table 4-2 Search Filter Boolean Operators

Operator	Symbol	Description
`AND`	&	All specified filters must be true for the statement to be true. For example, `(&(filter)(filter)(filter)...)`
`OR`	|	At least one specified filter must be true for the statement to be true. For example, `(|(filter)(filter)(filter)...)`
`NOT`	!	The specified statement must not be true for the statement to be true. Only one filter is affected by the `NOT` operator. For example, `(!(filter))`

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first
- All expressions from left to right

Specifying Search Filters Using a File

You can enter search filters into a file instead of entering them on the command line. When you do this, specify each search filter on a separate line in the file. The `ldapsearch` command runs each search in the order in which it appears in the file.

For example, if the file contains:

```
sn=Daniels
givenname=Charlene
```

then `ldapsearch` first finds all the entries with the surname Daniels, and then all the entries with the givenname Charlene. If an entry is found that matches both search criteria, the entry is returned twice.

For example, suppose you specified the previous search filters in a file named `searchdb`, and you set your search base using `LDAP_BASEDN`. The following returns all the entries that match either search filter:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -f searchdb
```

You can limit the set of attributes returned here by specifying the attribute names that you want at the end of the search line. For example, the following `ldapsearch` command performs both searches, but returns only the DN and the `givenname` and `sn` attributes of each entry:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -f searchdb
sn givenname
```

Specifying DNs that Contain Commas in Search Filters

When a DN within a search filter contains a comma as part of its value, you must escape the comma with a backslash (\). For example, to find everyone in the example.com Bolivia, S.A. subtree, use the following command:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -s base -b
"o=example.com Bolivia\, S.A.,dc=example,dc=com" "objectclass=*"
```

Using Client Authentication When Searching

This example shows user `cdaniels` searching the directory using client authentication:

```
ldapsearch -h myServer -p 636 -b "dc=example,dc=com" -N "cdanielsscertname" -Z -W
certdbpassword -P /home/cdaniels/certdb/cert.db "givenname=Richard"
```

NOTE The `ldapsearch` utility described in this section is not the utility provided with the Solaris platform, but is part of the Sun ONE Directory Server Resource Kit. For more information on this utility, refer to the *Sun ONE Directory Server Resource Kit Tools Reference*.

Search Filter Examples

The following filter searches for entries containing one or more values for the manager attribute. This is also known as a presence search:

```
manager=*
```

The following filter searches for entries containing the common name Ray Kultgen. This is also known as an equality search:

```
cn=Ray Kultgen
```

The following filter returns all entries that do not contain the common name Ray Kultgen:

```
(!(cn=Ray Kultgen))
```

The following filter returns all entries that contain a description attribute that contains the substring `X.500`:

```
description=*X.500*
```

The following filter returns all entries whose organizational unit is Marketing and whose description field does not contain the substring `X.500`:

```
(&(ou=Marketing)(!(description=*X.500*)))
```

The following filter returns all entries whose organizational unit is Marketing and that have Julie Fulmer or Cindy Zwaska as a manager:

```
(&(ou=Marketing)(|(manager=cn=Julie Fulmer,ou=Marketing,dc=example,
dc=com)(manager=cn=Cindy Zwaska,ou=Marketing,dc=example,dc=com)))
```

The following filter returns all entries that do not represent a person:

```
(!(objectClass=person))
```

The following filter returns all entries that do not represent a person and whose common name is similar to `printer3b`:

```
(&(!(objectClass=person))(cn~=printer3b))
```

Searching for Operational Attributes

If you want operational attributes returned as a result of a search operation, you must explicitly specify them in the search command.

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password aci=accounts
```

To retrieve regular attributes in addition to explicitly specified operational attributes, specify “*” in addition to the operational attributes. For example:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password
"objectclass=*" aci=accounts
```

Adding, Changing and Deleting Entries

The `ldapmodify` and `ldapdelete` utilities enable you to provide LDIF input to the directory to add, change and delete entries. These utilities read any number of update statements from the standard input or from a file, and modify the corresponding entries according to the LDIF instructions.

An update statement contains the DN of the target entry for the update, the operation to perform, and any data for the entry’s attributes. The kind of operation that will be performed is specified by the `changetype` keyword.

`ldapmodify` and `ldapdelete` read the statements that you enter in exactly the same way as if they were read from a file. When you finish providing input, enter the character that your shell recognizes as the end of file (EOF) escape sequence. The utility then begins operations based on the input you supplied.

Typically, the EOF escape sequence is one of the following, depending upon the underlying operating system:

- UNIX—Almost always Ctrl-D (^D)
- Windows—Usually Ctrl-Z followed by a carriage return (^Z<return>)

Adding and Changing Entries Using ldapmodify

The `ldapmodify` command-line utility enables you to add and change directory entries. `ldapmodify` opens a connection to the specified server using the distinguished name and password you supply, and modifies the entries based on LDIF update statements contained in a specified file, or in standard input.

Adding an Entry

The following `ldapmodify` command and LDIF update statement add a user called Sylvia Garcia to the example.com database:

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
```

```
dn: cn=Sylvia Garcia,ou=People,o=MyCorp,dc=example,dc=com
changetype: add
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Sylvia Garcia
givenName: Sylvia
sn: Garcia
ou: People
o: MyCorp
uid: sgarcia
```

Once you have added the entry, use the `ldapsearch` command to check that the entry exists.

Adding an Attribute to an Entry

The following `ldapmodify` command and LDIF update statement change Sylvia Garcia's entry to include a new attribute, `telephonenumber`.

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
```

```
dn: cn=Sylvia Garcia,ou=People,o=MyCorp,dc=example,dc=com
changetype: modify
add: telephonenumber
telephonenumber: 5552714
```

Once you have changed the entry, use the `ldapsearch` command to check that the new attribute has been added.

Changing the Value of an Attribute

The following `ldapmodify` command and LDIF update statement change Sylvia Garcia's telephone number by changing the value of the `telephonenumber` attribute.

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
```

```
dn: cn=Sylvia Garcia,ou=People,o=MyCorp,dc=example,dc=com
changetype: modify
replace: telephonenumber
telephonenumber: 555-1214
```

Once you have changed the entry, use the `ldapsearch` command to check that the attribute value has been changed.

NOTE If you add a value that has a *trailing space*, `ldapsearch` base-64 encodes the value, in order to represent the value including the space. This is particularly problematic with configuration entries.

In the preceding example, change the value of the `telephonenumber` attribute to `555-1215[]` (where [ ] represents a blank space):

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
```

```
dn: cn=Sylvia Garcia,ou=People,o=MyCorp,dc=example,dc=com
changetype: modify
replace: telephonenumber
telephonenumber: 555-1215[]
```

Now, use the `ldapsearch` command to check the value of the attribute:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -b
"dc=example,dc=com" "givenname=Sylvia"
```

```
dn: cn=Sylvia Garcia,ou=People,o=MyCorp,dc=example,dc=com
uid: SGarcia
givenName: Sylvia
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
sn: Garcia
cn: Sylvia Garcia
userPassword: {SSHA}2YpgER/oquWQfyd/vsEnxYmK0BVBO+Bx8U12QA==
telephoneNumber:: NTU1LTQxMjcg
```

Note that the value of the `telephoneNumber` attribute has been base-64 encoded.

Deleting Entries Using ldapdelete

Use the `ldapdelete` command-line utility to delete entries from the directory. This utility opens a connection to the specified server using the distinguished name and password you provide, and deletes the entry or entries.

Deleting a User Entry

The following `ldapdelete` command deletes the user, Sylvia Garcia from the example.com database:

```
ldapdelete -h myServer -p 5201 -D "cn=directory manager" -w password "cn=Sylvia
Garcia,ou=People,o=MyCorp,dc=example,dc=com"
```

Working With the Schema

As stated previously, the schema defines the size, range and format of entries that are stored in the directory. The schema is stored as a number of LDIF files in the `slapd-machine name/config/schema/` directory. In our sample installation, the schema is located in `slapd-myServer/config/schema`. Have a look at this directory and note the various LDIF files that define the schema.

Looking at the Schema Entry

Use the following ldapsearch command to look at the entry `cn=schema`.

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -b cn=schema
"objectclass=*"
```

This entry determines the valid object classes, the required attributes for these object classes, and the optional attributes, of every entry in the directory.

Schema Violation

If you attempt to add an entry that does not conform to the schema, the directory issues an Object Class Violation error.

Try to execute the following `ldapmodify` command to add an entry for a new user, `Frank`:

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
```

```
dn: cn=Frank,ou=People,o=MyCorp,dc=example,dc=com
changetype: add
objectclass: top
objectclass: person
givenName: Frank
cn: Frank
mail: frank@mycorp.com
userPassword: myPassword
```

Note that the directory responds with an Object Class Violation error. Can you see why this entry violates the schema?

The reason for the violation can be found in the `errors` log, described later in this chapter. For now it is sufficient to know that the `person` objectclass requires the surname (`sn`) attribute.

Use the same `ldapmodify` command to add the entry again, with an `sn` attribute. Once you have added the entry, use the `ldapsearch` command to check that it has been added.

Working With Groups and Roles

Group definitions are special entries that either name their members in a static list (static groups) or provide a filter which defines a dynamic set of entries (dynamic groups). The scope of possible members of a group is the entire directory, regardless of where the group definition entries are located.

Roles are defined and administered like groups, but in addition, member entries also have a generated attribute that indicates the roles in which they participate. This means that you can search a member entry to obtain a list of the roles in which it participates. As described in the previous chapter, roles can be *managed*, *filtered* or *nested*.

Adding a Static Group

The entry that defines a static group inherits from the `groupOfUniqueNames` object class. The group members are listed by their DN as multiple values of the `uniqueMember` attribute.

In “Populating a New Database Using ldif2db,” on page 74, we created a database for example.com. This database had one group called Administrators, with two members. Use the following `ldapmodify` command to add a second static group called HR Managers, with one member, Charlene Daniels:

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
```

```
dn: cn=HR Managers,ou=Groups,o=MyCorp,dc=example,dc=com
changetype: add
objectclass: top
objectclass: groupOfUniqueNames
cn: HR Managers
uniqueMember: uid=cdaniels,ou=People,o=MyCorp,dc=example,dc=com
```

Once you have added the group, use the `ldapsearch` command to check that it has been added.

Adding a Dynamic Group

The entry that defines a dynamic group inherits from the `groupOfUniqueNames` and `groupOfURLs` object classes. Group membership is defined by the filter given in the `memberURL` attribute. The members in a dynamic group are the entries that match the filter whenever it is evaluated.

The following example adds a dynamic group called “Smith Team”. The members of this group include all include all employees whose manager is ASmith.

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
```

```
dn: cn=Smith Team,ou=Groups, dc=example,dc=com
changetype: add
objectClass: top
objectClass: groupOfUniquenames
objectClass: groupOfUrls
cn: Smith Team
memberURL: ldap:///dc=example,dc=com??sub?(&(|(objectclass=person)
 (objectclass=groupofuniquenames))(manager=ASmith))
```

Adding a Managed Role Definition

All role definitions inherit from the `ldapsubentry` and `nsRoleDefinition` object classes. In addition, managed roles inherit from the `nsSimpleRoleDefinition` and `nsManagedRoleDefinition` object classes. Managed roles have a role definition entry and members are designated adding the `nsRoleDN` attribute to each member entry.

The following example adds a managed role called “myCorpAdminRole”.

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
```

```
dn: cn=myCorpAdminRole,o=myCorp,dc=example,dc=com
changetype: add
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsSimpleRoleDefinition
objectclass: nsManagedRoleDefinition
cn: myCorpAdminRole
description: managed role for administrators
```

Members of a managed role have the `nsRoleDN` attribute in their entry. Assign the role created above to the user Aaron Brown by updating his entry with the following `ldapmodify` command:

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
```

```
dn: uid=abrown,ou=People,o=MyCorp,dc=example,dc=com
changetype: modify
add: nsRoleDN
nsRoleDN: cn=myCorpAdminRole,o=myCorp,dc=example,dc=com
```

Searching for Role Definitions

Roles are `ldapSubEntry` objects. Entries of the `ldapsubentry` object class are not returned in response to normal search requests. You must specify that you are searching for a subentry in the search filter.

Use the following `ldapsearch` command to locate the managed role definition created previously:

```
ldapsearch -h myServer -p 5201 -D "cn=directory manager" -w password -b
"dc=example,dc=com"
"(&(objectclass=nsManagedRoleDefinition)(objectclass=ldapSubEntry))"
```

Working With Class of Service (CoS)

Revise the instructions for using the Console in “Working With Class of Service (CoS),” on page 58. The following example sets up the same CoS entries, using the command line.

Adding a Pointer CoS Entry

All CoS definition entries inherit from the `ldapsubentry` object class and the `cosSuperDefinition` object class. In addition, pointer CoS entries require the following object classes and attributes:

```
objectclass: cosPointerDefinition
cosTemplateDN: DN_string
cosAttribute: list_of_attributes qualifier
```

When using pointer CoS, the template entry inherits from the `ldapsubentry` object class and is also an instance of the `cosTemplate` object class. This entry must be created specifically for the CoS definition.

The following `ldapmodify` command creates a pointer CoS that shares a common postal code with all entries in the dc=example,dc=com tree:

```
ldapmodify -h myServer -p 5201 -a -D "cn=directory manager" /
 -w password

dn: cn=pointerCoS,dc=example,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn:
cn=cosTemplateForPostalCode,cn=data,dc=example,dc=com
cosAttribute: postalCode

dn: cn=cosTemplateForPostalCode,cn=data,dc=example,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: extensibleobject
objectclass: cosTemplate
postalCode: 44438
```

Working With ACIs

ACIs enable you to control access to the entire directory, a subtree of the directory, specific entries in the directory (including entries defining configuration tasks), or a specific set of entry attributes. You can create ACIs manually using LDIF statements, and add them to the directory tree using `ldapmodify`.

The `aci` attribute uses the following syntax:

`aci: (`*target*`)(version 3.0;acl "`*name*`";`*permission bind_rules*`;)`

where:

- *target* specifies the entry, attributes, or set of entries and attributes for which you want to control access. The target can be a distinguished name, one or more attributes, or a single LDAP filter. The target is an optional part of the ACI.
- version 3.0 is a required string that identifies the ACI version.
- "*name*" is a name for the ACI. The name can be any string that identifies the ACI. The ACI name is required.
- *permission* specifically outlines what rights you are either allowing or denying (for example, read or search rights).
- *bind_rules* specify the credentials and bind parameters that a user has to provide to be granted access. Bind rules can also specifically deny access to certain users or groups of users.

Looking at Access Restrictions

Use the following `ldapsearch` command to search the directory as user Charlene Daniels:

```
ldapsearch -h myServer -p 5201 -D
"uid=cdaniels,ou=People,o=MyCorp,dc=example,dc=com" -w cdaniels -b
"dc=example,dc=com" "objectclass=*"
```

Note that no entries are returned. This is because user Charlene Daniels does not have the appropriate access rights to search the directory.

Adding an ACI at the Command-Line

Use the following `ldapmodify` command to give Charlene Daniels full rights to the directory:

```
ldapmodify -h myServer -p 5201 -D "cn=directory manager" -w password
dn: o=MyCorp,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr=*)(version 3.0; aci "give charlene full rights";
allow(all) userdn =
 "ldap:///uid=cdaniels,ou=People,o=MyCorp,dc=example,dc=com";)
```

Once you have added the ACI, attempt the previous `ldapsearch` again. Note that all entries in the example.com database are now returned.

Examining the Log Information

Sun ONE Directory Server has several log files that contain detailed log information. The log files are stored in the the `slapd-`*serverID*`/logs` directory. In our installation it is located in `slapd-myServer/logs`.

Looking at the Access Log

The access log contains detailed information about client connections to the directory. To view the access log:

1. Change to *ServerRoot*`/slapd-myServer/logs`.
2. On UNIX systems, you can view the latest changes to the access log using the following command:

   ```
   tail -f access
   ```

3. Perform the following simple search and note the entry in the access log.

   ```
   ldapsearch -p 5201 -h myserver -D "cn=directory manager"
    -w password -b "dc=example,dc=com" "objectclass=*"
   ```

Enhancing the Error Log Information

You can specify the amount of log information that you want to display in the log files.

1. Change to *ServerRoot*`/slapd-myServer/logs`.
2. On UNIX systems, you can view the latest changes to the error log using the following command:

   ```
   tail -f errors
   ```

3. Use the following command to increase the error log level:

   ```
   ldapmodify -h myserver -p 5201 -D "cn=directory manager" -w
    password
   dn: cn=config
   changetype: modify
   replace: nsslapd-errorlog-level
   nsslapd-errorlog-level: 65573
   ```

4. On UNIX sytems, you can use the following command to view the level of information now being generated in the error log:

   ```
   tail -f errors
   ```

Searching an Internationalized Directory

When you perform search operations, you can request that the directory sort the results based on any language for which the server has a supporting *collation order*. For a list of the collation orders supported by the directory, see “Identifying Supported Locales” in the *Sun ONE Directory Server Reference Manual*.

This section focuses on the matching rule filter portion of the `ldapsearch` syntax. For more information on general `ldapsearch` syntax, see “LDAP Search Filters,” on page 81. For information on searching internationalized directories using the Users and Groups portion of the Sun ONE Server Console, refer to the online help or *Sun ONE Server Console Server Management Guide*.

This section covers the following topics:

- Matching Rule Filter Syntax
- Supported Search Types
- International Search Examples

Matching Rule Filter Syntax

A matching rule provides special guidelines for how the directory compares strings during a search operation. In an international search, the matching rule tells the system what collation order and operator to use when performing the search operation. The syntax of the matching rule filter is as follows:

attr`:`*matchingRule*`:=`*value*

where:

- *attr* is an attribute belonging to entries you’re searching for, such as `cn` or `mail`
- *matchingRule* is a string that identifies either the collation order or the collation order and a relational operator, depending on the format you prefer. For a discussion of matching rule formats, see “Matching Rule Formats,” on page 99.
- *value* is either the attribute value for which you want to search or a relational operator plus the attribute value for which you want to search. The syntax of the value portion of the filter depends on the matching rule format you use.

Matching Rule Formats

The matching rule portion of a search filter can be represented in the following ways:

- As the OID of the collation order for the locale on which you want to base your search.
- As the language tag associated with the collation order on which you want to base your search.
- As the OID of the collation order and a suffix that represents a relational operator.
- As the language tag associated with the collation order and a suffix that represents a relational operator.

The syntax for each of these options is discussed in the following sections:

- Using an OID for the Matching Rule
- Using a Language Tag for the Matching Rule
- Using an OID and Suffix for the Matching Rule
- Using a Language Tag and Suffix for the Matching Rule

Using an OID for the Matching Rule

Each locale supported by Directory Server has an associated collation order OID. For a list of supported locales and their associated OIDs, see "Identifying Supported Locales" in the *Sun ONE Directory Server Reference Manual*.

You can use the collation order OID in the matching rule portion of the matching rule filter as follows:

attr`:`*OID*`:=(`*relational_operator value*`)`

The relational operator is included in the value portion of the string, separated from the value by a single space. For example, to search for all `departmentNumber` attributes that are at or after N4709 in the Swedish collation order, use the following filter:

```
departmentNumber:2.16.840.1.113730.3.3.2.46.1:=>= N4709
```

Using a Language Tag for the Matching Rule

Each locale supported by Directory Server has an associated language tag. For a list supported locales and their associated language tags, see “Identifying Supported Locales” in the *Sun ONE Directory Server Reference Manual*.

You can use the language tag in the matching rule portion of the matching rule filter as follows:

attr`:`*language-tag*`:=(`*relational_operator value*`)`

The relational operator is included in the value portion of the string, separated from the value by a single space. For example, to search the directory for all description attributes with a value of `estudiante` using the Spanish collation order, use the following filter:

```
cn:es:== estudiante
```

Using an OID and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, you can append a suffix that represents a specific operator to the OID in the matching rule portion of the filter. Combine the OID and suffix as follows:

attr`:`*OID+suffix*`:=`*value*

For example, to search for `businessCategory` attributes with the value `Softwareprodukte` in the German collation order, use the following filter:

```
businessCategory:2.16.840.1.113730.3.3.2.7.1.3:=Softwareprodukte
```

The **`.3`** in the previous example is the equality suffix.

Using a Language Tag and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, you can append a suffix that represents a specific operator to the language tag in the matching rule portion of the filter. Combine the language tag and suffix as follows:

attr`:`*language-tag+suffix*`:=`*value*

For example, to search for all surnames that come at or after `La Salle` in the French collation order, use the following filter:

```
sn:fr.4:=La Salle
```

Using Wildcards in Matching Rule Filters

When performing a substring search using a matching rule filter, you can use the asterisk (*) character as a wildcard to represent zero or more characters.

For example, to search for an attribute value that starts with the letter `k` and ends with the letter `n`, you would enter a `k*n` in the value portion of the search filter. Similarly, to search for all attribute values beginning with the letter `u`, you would enter a value of `u*` in the value portion of the search filter.

To search for a value that contains the asterisk (*) character, you must escape the * with the designated escape sequence, \5c2a. For example, to search for all employees with `businessCategory` attribute values of `Example*Net product line`, enter the following value in the search filter:

```
Example \2a*Net product line
```

Supported Search Types

The directory server supports the following types of international searches:

- equality (=)
- substring (*)
- greater than (>)
- greater than or equal to (>=)
- less than (<)
- less than or equal to (<=)

Approximate, or phonetic, and presence searches are supported only in English.

As with a regular `ldapsearch` search operation, an international search uses operators to define the type of search. However, when invoking an international search, you can either use the standard operators (=, >=, >, <, <=) in the value portion of the search string, or you can use a special type of operator, called a suffix (not to be confused with the directory suffix), in the matching rule portion of the filter. Table 4-3 on page 102 summarizes each type of search, the operator, and the equivalent suffix.

Table 4-3 Search Types, Operators, and Suffixes

Search Type	Operator	Suffix
Less than	`<`	`.1`
Less than or equal to	`<=`	`.2`
Equality	`=`	`.3`
Greater than or equal to	`>=`	`.4`
Greater than	`>`	`.5`
Substring	`*`	`.6`

International Search Examples

The following sections show examples of how to perform international searches on directory data. Each example gives all the possible matching rule filter formats so that you can become familiar with the formats and select the one that works best for you.

Less Than Example

When you perform a locale-specific search using the less than operator (<) or suffix (.1), you search for all attribute values that come before the given attribute in a specific collation order.

For example, to search for all surnames that come before the surname *Marquez* in the Spanish collation order, you could use any of the following matching rule filters:

```
sn:2.16.840.1.113730.3.3.2.15.1:=< Marquez
sn:es:=< Marquez
sn:2.16.840.1.113730.3.3.2.15.1.1:=Marquez
sn:es.1:=Marquez
```

Less Than or Equal to Example

When you perform a locale-specific search using the less than or equal to operator (<=) or suffix (.2), you search for all attribute values that come at or before the given attribute in a specific collation order.

For example, to search for all room numbers that come at or before room number *CZ422* in the Hungarian collation order, you could use any of the following matching rule filters:

```
roomNumber:2.16.840.1.113730.3.3.2.23.1:=<= CZ422
roomNumber:hu:=<= CZ422
roomNumber:2.16.840.1.113730.3.3.2.23.1.2:=CZ422
roomNumber:hu.2:=CZ422
```

Equality Example

When you perform a locale-specific search using the equal to operator (=) or suffix (.3), you search for all attribute values that match the given attribute in a specific collation order.

For example, to search for all `businessCategory` attributes with the value `Softwareprodukte` in the German collation order, you could use any of the following matching rule filters:

```
businessCategory:2.16.840.1.113730.3.3.2.7.1:== Softwareprodukte
businessCategory:de:== Softwareprodukte
businessCategory:2.16.840.1.113730.3.3.2.7.1.3:=Softwareprodukte
businessCategory:de.3:=Softwareprodukte
```

Greater Than or Equal to Example

When you perform a locale-specific search using the greater than or equal to operator (>=) or suffix (.4), you search for all attribute values that come at or after the given attribute in a specific collation order.

For example, to search for all localities that come at or after *Québec* in the French collation order, you could use any of the following matching rule filters:

```
locality:2.16.840.1.113730.3.3.2.18.1:=>= Québec
locality:fr:=>= Québec
locality:2.16.840.1.113730.3.3.2.18.1.4:=Québec
locality:fr.4:=Québec
```

Greater Than Example

When you perform a locale-specific search using the greater than operator (>) or suffix (.5), you search for all attribute values that come at or before the given attribute in a specific collation order.

For example, to search for all mail hosts that come after host *schranka4* in the Czech collation order, you could use any of the following matching rule filters:

```
mailHost:2.16.840.1.113730.3.3.2.5.1:=> schranka4
mailHost:cs:=> schranka4
mailHost:2.16.840.1.113730.3.3.2.5.1.5:=schranka4
mailHost:cs.5:=schranka4
```

Substring Example

When you perform an international substring search, you search for all values that match the given pattern in the specified collation order.

For example, to search for all user IDs that end in *ming* in the Chinese collation order, you could use any of the following matching rule filters:

```
uid:2.16.840.1.113730.3.3.2.49.1:=* *ming
uid:zh:=* *ming
uid:2.16.840.1.113730.3.3.2.49.1.6:=* *ming
uid:zh.6:=* *ming
```

Appendix A

Accessibility Features

Based on the Java^{TM} Foundation Classes (JFC), the Sun ONE Directory Server console provides support for the assistive software and technologies that make software accessible to users with disabilities. This appendix describes the accessibility features of Sun ONE Directory Server Console, and the improvements that have been made to the document set to make it more accessible.

Console Accessibility Features

Most of the accessibility features described in the following section are provided automatically through the use of JFC / Swing components. For information on how to customize user settings, refer to the *Sun ONE Directory Server Administration Guide*.

Accessible Names and Descriptions

All objects have accessible names (succinct explanations of the object’s purpose). These names can be used by assistive technologies to present the objects to the user. Accessible descriptions are more verbose explanations that provide additional information on objects, where this is necessary.

Customizable Fonts

The style and size of fonts in text panes, menus, labels, and information messages, can be customized.

Although color coding is used to convey information, it is not the only means of doing so.

Dynamic GUI Layout

The dynamic layout allows users to specify the size and position of Directory Server windows, or for this to be determined by the user's settings.

Keyboard Traversable Components

This accessibility feature caters for users who have difficulty using a mouse. Pressing the tab key moves the input focus from component to component and shift-tab moves the focus in the opposite direction. The arrow keys allow users to navigate trees without using the mouse.

The focus is programmatically exposed so that assistive software can track focus and focus changes.

Text Equivalents For Non-Text Elements

When an image represents a program element, the information conveyed by the image is also available in the text.

Equivalent Command-Line Interface

Most of the functionality of the console can be achieved at the command-line. This command-line interface is comprehensively documented in the *Sun ONE Directory Server Reference Manual*.

Documentation Accessibility Features

The Sun ONE Directory Server 5.2 document set is delivered in both PDF and HTML format. This section describes accessibility features in the HTML version of the documentation.

Text Equivalents For Non-Text Elements

Alternative text labels are assigned to links or graphics. Where graphics provide detailed descriptions, text versions of these descriptions are provided either within the surrounding text, or in a separate file.

Tables That Can be Interpreted by Assistive Technology

All tables now include descriptive headers. A brief description of the table contents is also provided in the surrounding text.

Glossary

access control instruction (ACI) An instruction that grants or denies permissions to entries in the directory.

access control list (ACL) The mechanism for controlling access to your directory. In Sun ONE Directory Server, an ACL is an `aci` attribute in a directory entry.

access rights In the context of access control, specify the level of access granted or denied. Access rights are related to the type of operation that can be performed on the directory. The following rights can be granted or denied: read, write, add, delete, search, compare, selfwrite, proxy and all. For more information on these rights, see Chapter 6, "Managing Access Control", in the *Sun ONE Directory Server Administration Guide.*

account inactivation Disables a single user account, or set of accounts, so that all authentication attempts are automatically rejected.

All IDs Threshold A size limit which is globally applied to every index managed by the server. When the size of an *entry ID list* reaches this limit, the server replaces that entry ID list with an All IDs token.

All IDs token A mechanism which causes the server to assume that all directory entries match the index key. In effect, the All IDs token causes the server to perform an unindexed search to match the index key.

anonymous access When granted, allows anyone to access directory information without providing credentials, and regardless of the conditions of the bind.

approximate index Allows for efficient approximate or "sounds-like" searches.

attribute Holds descriptive information about an entry. Attributes have a type (name) and a set of values. An attribute type also specifies the syntax for the kind of information that can be stored as values of attributes of that type.

attribute list See *optional attribute list* and *required attribute list*.

authenticating Directory Server In pass-through authentication (PTA), the authenticating Directory Server contains the authentication credentials of the requesting client. A PTA-enabled user directory passes through bind requests to the authenticating directory, which verifies the bind credentials of the requesting client.

authentication The process of proving the identity of the client user to the Directory Server. Users must provide a bind DN and either the corresponding password or certificate in order to be granted access to the directory. Directory Server allows the user to perform functions or access files and directories based on the permissions granted to that user by the directory administrator. See also *server authentication*.

authentication certificate An X.509 digital certificate, issued by a Certification Authority, that is marked as suitable for use in authentication during an *SSL* connection setup.

base distinguished name (base DN) Base distinguished name. A search operation may be performed on just the entry identified by the base DN, the entries which are immediately subordinate to it, or to the entry and all entries below it in the *directory information tree*.

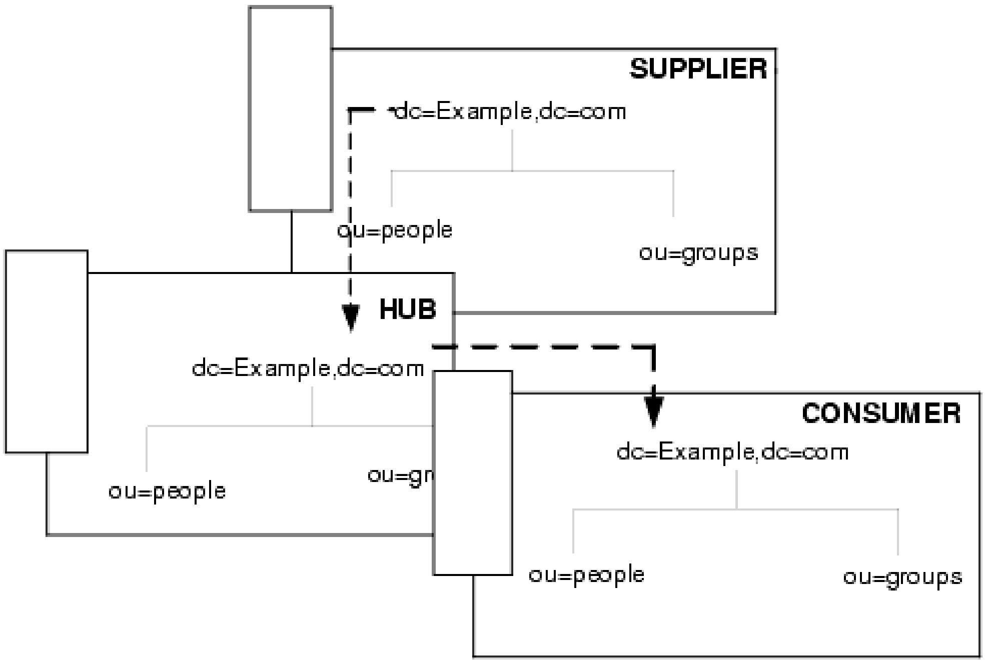
bind distinguished name (bind DN) Distinguished name used to authenticate to a Directory Server in the bind request.

bind rule In the context of access control, the bind rule specifies the credentials and conditions that a particular user or client must satisfy in order to get access to directory information.

browsing index Also known as a virtual list view (VLV) index. Speeds up the display of entries in the Directory Server Console (or other graphical user interface) if the client with the user interface uses the virtual list view extension. Virtual list view indexes can be created on any branch in the directory tree to improve display performance for specific searches.

CA *See Certificate Authority*.

cascading replication In a cascading replication scenario, one server, often called the hub supplier acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a change log. It receives updates from the supplier server that holds the master copy of the data, and in turn supplies those updates to the consumer, as shown in the following diagram.



certificate A certificate strongly associates the public key of a user or *CA* with the identity, typically a *distinguished name*, of that user or CA. The certificate is digitally signed by a Certificate Authority, and can be validated during an *SSL* connection setup to obtain the public key of the other end of the connection. X.509 certificates are stored within the directory in the `caCertificate;binary` or `userCertificate;binary` attributes.

Certificate Authority Company or organization that sells and issues authentication certificates. You may purchase an authentication certificate from a Certification Authority that you trust. Also known as a CA.

chained suffix An implementation of chaining. A chained suffix behaves like a normal suffix but has no persistent storage. Instead, it points to data stored remotely.

chaining A method for relaying requests to another server. Results for the request are collected, compiled and then returned to the client. In the context of replication, chaining occurs when a consumer replica receives an update request, and forwards it to the server that holds the corresponding master replica. Note that this is not the same as a *referral*.

change log A change log is a record of the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on consumer servers, or on other masters, in the case of multi-master replication. Note that this is not the same as the *retro changelog*, which is not used for replication.

character type Distinguishes alphabetic characters from numeric or other characters and the mapping of upper-case to lower-case letters.

ciphertext Encrypted information that cannot be read by anyone without the proper key to decrypt the information.

class definition Specifies the information needed to create an instance of a particular object.

class of service (CoS) A method for sharing attributes between entries.

classic CoS A classic CoS identifies the template entry by both its DN and the value of one of the target entry’s attributes.

client A software entity that requests services or information from a server.

collation order Provides language and cultural-specific information about how the characters of a given language are to be sorted. This information might include the sequence of letters in the alphabet or how to compare letters with accents to letters without accents.

computed attributes Attributes that are not stored with the entry itself but are returned to the client application along with normal attributes in operation results.

conflict A situation that arises when changes are made to the same directory data on different directory servers before replication can synchronize the data between the servers. When the servers do synchronize, they detect that their copies are inconsistent, and may resolve the conflict or log an error.

conflict resolution Deterministic procedures used to resolve change information. For more information, see “Solving Common Replication Conflicts” in the *Sun ONE Directory Server Administration Guide*.

consumer Server containing replicated directory trees or subtrees from a supplier server.

consumer replica A replica that refers all add, modify, and delete operations to master replicas. A server can hold any number of consumer replicas of different naming contexts.

container entry An entry that represents the top of a subtree in the directory.

CoS *See class of service.*

CoS definition entry Identifies the type of CoS you are using. It is stored as an LDAP subentry below the branch it affects.

CoS template entry Contains a list of the shared attribute values.

DAP Directory Access Protocol. The ISO / ITU-T X.500 protocol that was the basis for *LDAP*.

default index When Directory Server is installed, a set of default indexes is created for each database instance. For more information, see “Default Indexes” in the *Sun ONE Directory Server Administration Guide*.

definition entry See *CoS definition entry*.

Directory Access Protocol See *DAP*.

directory information tree The logical representation of the information stored in the directory. It mirrors the tree model used by most file systems, with the tree’s root point appearing at the top of the hierarchy. Also known as DIT.

Directory Manager The privileged database administrator, comparable to the root user in UNIX systems. Access control does not apply to the directory manager.

Directory Server Console An LDAP client application that provides a graphical user interface to browse, configure, and manage the contents of your directory. The Directory Server Console is a component of the Sun ONE Directory Server product.

directory service A database application designed to manage descriptive, attribute-based information about people and resources within an organization.

distinguished name String representation of an entry's name and location in the directory.

DIT See *directory information tree.*

DN See *distinguished name.*

DNS Domain Name System. The system used by machines on a network to associate IP addresses (such as 64.124.140.181) with hostnames (such as `www.sun.com`). Clients usually use DNS to find the IP addresses of servers they wish to contact. The data in DNS is often augmented in local tables, such as from NIS or the `/etc/hosts` file on UNIX systems.

DNS alias A DNS alias is a hostname that the *DNS* server knows points to a different host. The DNS alias is implemented as a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, an alias such as `www.[yourdomain].[domain]` might point to a real machine called `realthing.[yourdomain].[domain]` where the server currently exists.

DSA Directory System Agent (an X.500 term for a Directory Server).

DSE A DSE, or *DSA*-specific entry, has additional server-specific information associated with it. Some DSE's such as the *Root DSE* or schema DSE, have different attributes on each server.

DSML (Directory Services Markup Language). A family of document formats for representing XML markup language that enables you to represent directory services in XML. Sun ONE Directory Server 5.2 conforms to version 2 of the DSML standard (DSMLv2).

entry A group of attributes and a unique distinguished name.

entry distribution Method of distributing directory entries across more than one server in order to scale to support large numbers of entries.

entry ID list Each index that the directory uses is composed of a table of index keys and matching entry ID lists. The entry ID list is used by the directory to build a list of candidate entries that may match the client application's search request.

equality index Allows you to search efficiently for entries containing a specific attribute value.

filter The filter in a search request specifies a pattern which an entry in the scope of the search must match for that entry to be returned in the search response. Filters are also used in constructing role and access control definitions.

filtered role Allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

fractional replication Replication of a filtered subset of attributes.

general access When granted, indicates that all authenticated users can access directory information.

HTTP Hypertext Transfer Protocol

The method for exchanging information between HTTP servers and clients.

HTTPD An abbreviation for the *HTTP* daemon or service, a program that serves information using the HTTP protocol.

HTTPS Used in URLs to indicate that *HTTP* is layered on the Secure Sockets Layer, *SSL,* and so is protected against eavesdropping while in transit.

hub supplier In the context of replication, a server that holds a replica that is copied from a different server, and in turn replicates it to a third server. See also *cascading replication*.

immediate subordinate In the *DIT*, an *entry* is an immediate subordinate of another if its *distinguished name* is formed by appending its *RDN* to the distinguished name of the parent entry.

immediate superior In the *DIT*, an *entry* is the immediate superior of another if its *distinguished name,* followed by the *RDN* of the other entry, forms the distinguished name of the child entry.

index key Each index that the directory uses is composed of a table of index keys and matching entry ID lists.

indirect CoS An indirect CoS identifies the template entry using the value of one of the target entry's attributes.

international index Speeds up searches for information in a *directory information tree* in which the attributes have language tags.

LDAP Lightweight Directory Access Protocol. Directory service protocol designed to run over TCP/IP and across multiple platforms.

LDAPv3 Version 3 of the LDAP protocol.

LDAP URL Provides the means of locating directory servers using DNS and then completing the query via LDAP. A sample LDAP URL is `ldap://ldap.sun.com`

LDBM database A high-performance, disk-based database consisting of a set of large files that contain all of the data in Directory Server.

LDIF LDAP Data Interchange Format. Format used to represent Directory Server entries a text form using "`type: value`" pairs.

leaf entry An entry under which there are no other entries. A leaf entry cannot be a branch point in a directory tree.

Lightweight Directory Access Protocol *See LDAP.*

locale Identifies the collation order, character type, monetary format and date/time format used to present data for users of a specific region, culture, or custom. This includes information on how data of a given language is interpreted, stored, or collated. The locale also indicates which code page should be used to represent a given language. Note that in LDAP all attributes are in UTF-8.

managed object An SNMP data element that forms part of an *MIB*. In Sun ONE Directory Server, the managed objects are held in `cn=monitor`, and the SNMP agent provides to the *network management station*. As with LDAP attributes, each managed object has a name and object identifier expressed in dot-notation.

managed role Allow you to create an explicit enumerated list of members.

management information base *See MIB.*

mapping tree A data structure that associates the names of suffixes (subtrees) with databases.

master agent *See SNMP master agent.*

matching rule Provides guidelines for how the server compares strings during a search operation. In an international search, the matching rule tells the server what collation order and operator to use.

MD5 A message digest algorithm by RSA Data Security, Inc., which can be used to produce a short digest of data, that is unique with high probability, and is mathematically extremely hard to produce a piece of data that will produce the same message digest.

MD5 signature A message digest produced by the MD5 algorithm.

MIB Management Information Base. The collection of managed objects held by an SNMP agent.

multi-master replication A replication model in which entries can be written and updated on any of several master replica copies without requiring communication with other master replicas before the write or update is performed. Each server maintains a *change log* for the replica. Modifications made on one server are automatically replicated to the other servers. In case of *conflict*, a time stamp is used to determine which server holds the most recent version.

multiplexor The server containing the database link that communicates with the remote server.

n + 1 directory problem The problem of managing multiple instances of the same information in directories and databases of different types, resulting in increased hardware and personnel costs.

name collision A conflict that occurs during replication if multiple entries have been added or renamed, and attempt to use the same distinguished name. The conflicting entries are renamed automatically by the directory servers to ensure DN uniqueness.

nested role A role that names other role definitions. The set of members of a nested role is the union of all members of the roles it contains. Nested roles may also define extended scope to include the members of roles in other subtrees

network management station Powerful workstation with one or more network management applications installed.

NIS Network Information Service. A system of programs and data files that UNIX systems use to collect, collate, and share specific information about machines, users, file systems, and network parameters throughout a network of computers.

ns-slapd On UNIX systems, this is the process or service responsible for all actions of the Directory Server. On Windows systems, the equivalent is *slapd.exe*.

ns-slapd.exe The `slapd` process watchdog on Windows systems.

object class Defines an entry type in the directory by defining which attributes are contained in the entry.

object identifier (OID) A string representation of an object identifier consists of a list of decimal numbers separated by periods, e.g. "1.3.6.1.4.1". In LDAP, object identifiers are used to uniquely identify schema elements, including object classes and attribute types. The top levels of an object identifier hierarchy are managed by standards bodies and are delegated to organizations who wish to construct their own schema definitions.

operational attribute An operational attribute contains information used internally by the directory to keep track of modifications and subtree properties. Operational attributes are not returned in response to a search unless explicitly requested.

optional attribute list A list of optional attributes for a specified object class. Optional attributes are preceeded by the keyword MAY.

parent access When granted, indicates that users have access to entries below their own in the directory tree, that is, if the bind DN is the parent of the targeted entry.

pass-through authentication *See PTA.*

pass-through subtree In pass-through authentication, the *PTA Directory Server* will pass through bind requests to the *authenticating Directory Server* from all clients whose DN is contained in this subtree.

password policy A set of rules that govern how passwords are used in a given directory.

permission In the context of access control, the permission states whether access to the directory information is granted or denied, and the level of access that is granted or denied. See also *access rights*.

pointer CoS A pointer CoS identifies the template entry using the template DN only.

presence index Enables efficient searching for entries that contain an attribute of a specified type, regardless of the value of the attribute in the entry.

propagation behaviour The synchronization process between a *consumer* and a *supplier*.

protocol A set of rules that describes how devices on a network exchange information.

proxy authorization A special form of authentication where a user binds to the directory with its own identity but is granted the access rights of another user on a per operation basis. This other user is referred to as the proxy user, and its DN the proxy DN.

proxy DN Used with proxied authentication. The proxy DN is the DN of an entry that has access permissions to the target on which the client application is attempting to perform an operation.

PTA Pass-through authentication. Mechanism by which one Directory Server consults another to check bind credentials.

PTA Directory Server In pass-through authentication (PTA), the PTA Directory Server sends (passes through) bind requests it receives to the authenticating Directory Server.

PTA LDAP URL In pass-through authentication, the URL that defines the *authenticating Directory Server*, *pass-through subtree*(s) and optional parameters.

RDN Relative distinguished name. The name of the actual entry itself, before the entry's ancestors have been appended to the string to form the full distinguished name. Most RDNs consist of a single attribute type and value from the entry.

referential integrity Mechanism that ensures that relationships between entries expressed by DN-valued attributes are maintained within the directory.

referral When a server receives a search or update request from a client that it cannot process, it sends back to the client a pointer to the Directory Server that can process the request.

referral hop limit The maximum number of *referral*s that a client should follow in a row.

relative distinguished name *See RDN.*

replica An instance of an area of *replication* on a server. See also *consumer replica* and *supplier replica.*

replica cycle See *replication cycle.*

replica group The servers that hold instances of a particular area of *replication*. A server may be part of several replica groups.

replication The process of synchronizing data distributed across Directory Servers and rectifying update *conflict*s.

replication agreement Set of configuration parameters that are stored on the supplier server and identify the suffixes to replicate, the consumer servers to which the data is pushed, the times during which replication can occur, the DN and credentials used by the supplier to bind to the consumer, and how the connection is secured.

replication base entry The DN of the root of a replicated area.

replication cycle The interval during which update information is exchanged between two or more replicas. The replication cycle begins during an attempt to push data to, or pull data from, another replica or set of replicas, and ends when the data has successfully been exchanged or when an error is encountered.

replication session A session set up between two servers in a replica group to pass update information as part of a replica cycle.

required attribute list A list of required attributes for a specified object class. Required attributes are preceeded by the keyword MUST.

retro changelog Provides backward compatibility with 4.x releases of Directory Server. The retro changelog stores changes in the order of arrival on the local server and not in the order in which these changes were applied to the system. The retro changelog was not designed to function in a multi-master replication environment. Note that this is not the same as the *change log*, as the retro changelog is not used in replication.

role An entry grouping mechanism. Each role has *members*, which are the entries that possess the role.

role-based attributes Attributes that appear on an entry because it possesses a particular role within an associated CoS template.

root DN The distinguished name of the Directory Manager (the superuser for the Directory Server).

Root DSE An entry that is automatically generated by the Directory Server and is returned from a `baseObject` search with a DN that is empty (zero bytes long). The Root DSE provides information to clients about the server's configuration, such as a pointer to the subschema entry, a list of the DNs of the naming contexts held by the server, and a list of the LDAPv3 controls and extensions which the server supports. See also *DSE*.

root suffix The parent of one or more sub suffixes. A directory tree can contain more than one root suffix.

RTT Round Trip Time (also called the round-trip delay time). The elapsed time for transit of a signal over a closed circuit (from the server to the client and back). This delay is important in systems that require two-way interactive communication where the RTT directly affects the throughput rate. In the context of Sun ONE Directory Server, the RTT and the TCP window can have a significant impact on *replication* performance over WAN.

schema Definitions describing what types of information can be stored as entries in the directory. When information that does not match the schema is stored in the directory, clients attempting to access the directory may be unable to display the proper results.

schema checking Ensures that entries added or modified in the directory conform to the defined schema. Schema checking is on by default and users will receive an error if they try to save an entry that does not conform to the schema.

Secure Sockets Layer *See SSL.*

self access When granted, indicates that users have access to their own entries, that is, if the bind DN matches the targeted entry.

server authentication Allows a client to make sure that it is connected to a secure server, preventing another computer from impersonating the server or attempting to appear secure when it is not.

server root Also known as *ServerRoot*. A directory on the server machine dedicated to holding the server program and configuration, maintenance, and information files.

Simple Network Management Protocol *See SNMP.*

single-master replication A *replication* model in which only one server, the master, allows LDAP write access to the replicated data. In a single-master replication model, the supplier or master server maintains a *change log*.

slapd.exe On Windows systems, this is the process or service responsible for all actions of the Directory Server. On UNIX systems, the equivalent is *ns-slapd*.

SNMP Simple Network Management Protocol. Used to monitor and manage application processes running on the servers, by exchanging data about network activity.

SNMP master agent Software that exchanges information between the various subagents and the NMS.

SNMP subagent Software that gathers information about the managed device and passes the information to the master agent.

SOAP Simple Object Access Protocol. A lightweight, *XML*-based protocol for the exchange of information in a decentralized, distributed environment. In the context of Sun ONE Directory Server, SOAP is used with HTTP to provide a framework for describing the contents of messages and how to process them.

SSL Secure Sockets Layer. A software library establishing a secure connection between two parties (client and server) used to implement *HTTPS*, the secure version of *HTTP*.

standard index Indexes that are maintained by default.

subagent *See SNMP subagent*.

subschema entry An entry containing all the schema definitions (definitions of object classes, attributes, matching rules, and so on) used by entries in part of a directory tree.

substring index Allows for efficient searching against substrings within entries. Substring indexes are limited to a maximum of three characters per *index key*.

sub suffix A branch underneath a root suffix.

suffix The name of the entry in the directory tree, below which data is stored. Multiple suffixes are possible within the same directory. Each database only has one suffix.

supplier Server containing the master copy of directory trees or subtrees that are replicated to consumer servers.

supplier replica A replica that contains a master copy of directory information and can be updated. A server can hold any number of master replicas.

supplier server In the context of replication, a server that holds a replica that is copied to a different server is called a supplier for that replica.

symmetric encryption Encryption that uses the same key for both encrypting and decrypting. The Data Encryption Standard (DES) is an example of a symmetric encryption algorithm.

system index An index that cannot be deleted or modified as it is essential to Directory Server operations.

target In the context of access control, the target identifies the directory information to which a particular ACI applies.

target entries The entries within the scope of a CoS.

TCP/IP Transmission Control Protocol/Internet Protocol. The main network protocols for the Internet and for enterprise networks.

template entry *See CoS template entry*.

TLS Transport Layer Security. The standard for Secure Socket Layers (SSL), a public key based protocol.

topology The way a directory tree is divided among physical servers and how these servers link with one another.

Transport Layer Security *See TLS.*

virtual list view index (VLV) See *browsing index*.

X.500 standard The set of ISO/ITU-T documents outlining the recommended information model, object classes and attributes used by Directory Server implementation. *LDAP* is a "lightweight" version of the Directory Access Protocol (DAP) used by the X.500 standard.

XML Extensible Markup Language. XML is an underlying format for structured textual document exchange on the web.

Index