Tools Reference

Sun™ ONE Directory Server Resource Kit

Version 5.2

816-6400-10
March, 2004
2nd Edition

Contents

List of Figures

List of Tables

List of Procedures

List of Code Examples

About This Guide

The Sun™ ONE Directory Server Resource Kit *Tools Reference* covers the installation of the Sun ONE Directory Server Resource Kit (DSRK) and details information about its command-line tools and included Lightweight Directory Access Protocol (LDAP) application programming interfaces (API). This preface contains the following sections:

- Audience for This Guide
- Directory Server Resource Kit Documentation Set
- Documentation Conventions Used in This Guide
- Related Sources of Information
- Acknowledgements
- Documentation Feedback

Audience for This Guide

This *Tools Reference* is intended for use by IT administrators and software developers who implement a centralized and distributed LDAP data repository for use in an intranet, an extranet for communication with your business partners, or over the public Internet to reach your customers. The DSRK is a set of tools for testing and maintaining LDAP directory servers. Therefore, they should understand how to administer the directory server and its contents. This includes basic directory architectural concepts needed to successfully design and deploy a directory service. It is also recommended that administrators understand the following technologies:

- LDAP
- LDAP Data Interchange Format (LDIF)

- Java™
- JavaServer Pages™
- Java Naming and Directory Interface (JNDI)
- HyperText Markup Language (HTML)
- eXtensible Markup Language (XML)

In addition, they should be familiar with Sun ONE Directory Server, Sun Microsystems' LDAP data repository, and the command-line syntax for the particular shell and platform on which the server is installed. The books and guides described in "Related Sources of Information" on page 31 provide a good introduction to this information.

NOTE This reference guide documents only the tools of the DSRK. It is not intended to be a deployment, testing, or maintenance guide for your directory installation. You should be aware of the following characteristics of your directory when using these tools:

- Size and number of entries.
- Directory structure and access permissions.
- Virtual attributes, class of service, and indexing.
- Usage, types of access, and access patterns.

Please refer to the *Sun ONE Directory Server Deployment Guide* for information on how to determine the influence of these factors and take them into consideration.

Directory Server Resource Kit Documentation Set

The DSRK documentation set contains three manuals:

- *Tools Reference*
- *Sun ONE LDAP SDK for C Programming Guide*
- *Sun ONE LDAP SDK for Java Programming Guide*

NOTE The LDAP API are part of the DSRK installation but are documented separately.

Tools Reference

The *Tools Reference* explains how to use each of the DSRK tools by detailing its command-line syntax, listing all options, their functionality, and all return values (where applicable), and providing examples of usage.

- Part 1, “Getting Started”provides an introduction to the tools and installation instructions for the product.
- Part 2, “Directory Access Tools” contains information on tools for accessing a LDAP directory. Use these commands to retrieve entries, view their attributes, and make modifications.
- Part 3, “Performance Tools” explains the tools that help you run tests to measure your server’s average response time. These tools perform repeated LDAP authentication, search, add, and delete operations to simulate actual usage.
- Part 4, “LDAP v3 Tools” explains the tools that are specific to directory server’s whose schema format is based on version 3 of the LDAP protocol (LDAPv3). These tool measure the performance of add, modify and search operations, and rate of authentication.
- Part 5, “LDIF Deployment Tools”describes the tools that process large LDIF (LDAP Data Interchange Format) files, either generating, modifying, or comparing the LDAP entries and attribute values they contain.
- Part 6, “Maintenance and Debugging Tools” illustrates the maintenance and debugging tools that help directory administrators to interpret logs and other trouble-shooting files.
- Part 7, “Sample Phonebook Application,”documents a sample application that uses servlets to provide a web interface to access a directory server.
- Part 8, “Additional Tools and Information,” contains the documentation for an unsupported plug-in module for Sun ONE Directory Server 5.2.

Sun ONE LDAP SDK for C Programming Guide

The *Sun ONE LDAP SDK for C Programming Guide* documents the Sun ONE LDAP software development kit (SDK) for C. The SDK includes C libraries which are used for writing LDAP client applications. The applications can be used to connect to, search, and update LDAP servers located on a network or on the Internet.

Sun ONE LDAP SDK for Java Programming Guide

The *Sun ONE LDAP SDK for Java Programming Guide* documents the Sun ONE LDAP SDK for Java. The SDK includes the interfaces, classes, and methods used to interact and communicate with LDAP servers. Also available are the Java API specifications or Javadocs.

Documentation Conventions Used in This Guide

In the Directory Server Resource Kit documentation, certain typographical conventions and terminology are used. These conventions are described in the following sections.

Typographical Conventions

This book uses the following typographical conventions:

- *Italic type* is used within text for book titles, new terminology, emphasis, and words used in the literal sense.
- `Monospace font` is used for sample code and code listings, API and language elements (such as function names and class names), filenames, pathnames, directory names, HTML tags, and any text that must be typed on the screen.
- *Italic serif font* is used within code and code fragments to indicate variable placeholders. For example, the following command uses *filename* as a variable placeholder for an argument to the `gunzip` command:

  `gunzip -d` *filename*`.tar.gz`

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

Terminology

Below is a list of general terms used in the Directory Server Resource Kit documentation set:

- *DSRK* refers to the installed Directory Server Resource Kit software.
- *Directory Server* refers to an installed instance of Sun ONE Directory Server.

- *DSRK_base* is a variable place holder for the home directory where Directory Server Resource Kit is installed.
- *DirectoryServer_base* is a variable place holder for the home directory where Sun ONE Directory Server is installed.

Related Sources of Information

In addition to the documentation provided with Directory Server Resource Kit, there are several other sources of information that might be helpful.

Directory Server Documentation

The tools in the DSRK interact with Sun ONE Directory Server. The following documentation is referenced throughout this guide:

- *Sun ONE Directory Server Deployment Guide*
- *Sun ONE Directory Server Installation and Tuning Guide*
- *Sun ONE Directory Server Administration Guide*
- *Sun ONE Directory Server Reference Manual*

A listing of the full set of documentation can be found at:

`http://docs.sun.com/coll/S1_DirectoryServer_52`

Sources of Information

Sun Microsystems has other helpful sources of information. Table 0-1 lists these sources.

Table 0-1 Additional Sources of Information

Source	Location
Sun ONE Download Center	`http://wwws.sun.com/software/download/`
Sun ONE Technical Support	`http://www.sun.com/service/sunone/software/index.html`
Sun ONE Professional Services Information	`http://www.sun.com/service/sunps/sunone/index.html`

Table 0-1 Additional Sources of Information *(Continued)*

Source	Location
Sun Enterprise Services, Solaris Patches and Support	http://sunsolve.sun.com/
JavaServer Pages	http://java.sun.com/products/jsp/
The Source for Developers	http://developers.sun.com/
Sun Training and Certification	http://suned.sun.com/

Third-Party Sources of Information

The following sections list sources of information found on sites that are not under the auspices of Sun Microsystems.

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

The Mozilla Project

The Sun ONE LDAP SDK for Java and Sun ONE LDAP SDK for C are also released through the Mozilla™ open source project. Instructions on obtaining the source code and contributing to the development of these API is available at:

```
http://www.mozilla.org/directory
```

Perl Development

Many of the tools in the DSRK are Perl scripts that rely on a Perl interpreter. Information about the Perl language and other Perl resources is available at:

- The Perl Directory: http://www.perl.org
- Comprehensive Perl Archive Network: http://www.cpan.org/
- Perl documentation: http://www.perldoc.com/

Tomcat Web Server

The new web-based gateway demonstrated in the sample phonebook application relies on the Tomcat application of the Jakarta project from the Apache Software Foundation:

`http://jakarta.apache.org/tomcat/tomcat-4.0-doc/index.html`

Acknowledgements

This product includes software developed by the Apache Software Foundation (`http://www.apache.org/`).

This product also includes software developed through Mozilla (`http://www.mozilla.org/`).

Documentation Feedback

Sun is interested in improving its documentation and welcomes your comments and suggestions. Use this web-based form to provide feedback to Sun:

`http://www.sun.com/hwdocs/feedback/`

Please provide the full document title and part number in the appropriate fields. The part number can be found on the title page of the book or at the top of the document, and is usually a seven or nine digit number. For example, the part number of this Directory Server Resource Kit 5.2 Tools Reference is 816-6400-10.

Part 1

Getting Started

Chapter 1

Introduction

The *Sun™ ONE Directory Server Resource Kit Tools Reference* provides installation instructions for the Sun ONE Directory Server Resource Kit as well as usage information for the included engineering tools and application programming interfaces (API). This introductory chapter contains the following sections:

- Overview
- Tools
- Sun ONE LDAP SDK for C
- Sun ONE LDAP SDK for Java

Overview

The Directory Server Resource Kit (DSRK) provides tools and API for deploying, accessing, tuning, and maintaining an installation of Sun ONE Directory Server. These utilities will help to implement and maintain a more robust server solution. The DSRK is comprised of three components:

- Tools
- Sun ONE LDAP SDK for C
- Sun ONE LDAP SDK for Java

The DSRK installation contains all the executables for running the tools as well as all the libraries needed for using the included LDAP SDKs (Software Development Kits).

The command-line tools and applications will help you test the performance of your Directory Server, and administer the contents of your directory. These tools are themselves based on the LDAP SDKs, and they were created to help Sun ONE development teams to test and validate the Sun ONE Directory Server.

The LDAP SDKs for C and Java™ make it simple to write client applications for your directory. These API expose all of 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.

NOTE Only the sample phone book application (discussed in Chapter 32, "JSP Directory Gateway Phonebook") requires the installation of further components. These components are also provided with the product and discussed in the aforementioned chapter.

Tools

The DSRK contains a set of tools that make a LDAP directory accessible using a command-line shell. This wide range of tools (including a sample application, Perl scripts, tag libraries and an LDAP command-line) can be used for directory access, performance testing, and maintenance. In addition, the commands that run these tools can be used to write scripts to automate the tasks. The following sections list the tools shipped with, and documented in, the DSRK. The tools are grouped within the book by functionality.

NOTE Many of the tools of the DSRK are command-line utilities whose functionality is available through options and parameters. You should be familiar with the syntax of commands for your particular shell and platform. For example, you may need to specify values that contain characters with special meaning to the command-line interpreter, such as space `" "`, asterisk `"*"`, backslash `"\"`, and so forth. Most shells require you to use quotation marks around values that contain special characters. One example is the space character in DNs (distinguished names):

```
"cn=Barbara Jensen,ou=Product Development,dc=siroe,dc=com"
```

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

The command examples given in this book are valid for most UNIX® shell environments. They rely on basic shell syntax and are usually applicable to all platforms with little modification.

Directory Access

The directory access commands provide the fundamental tools for accessing a LDAP directory. Use these commands to retrieve entries, view their attributes, and make modifications. These tools are based upon the Sun ONE LDAP SDK for C and make the functionality of this API available through their various options.

Table 1-1 Directory Access Commands

Tool	Chapter	Purpose
`ldapsearch`	The ldapsearch Tool	Perform simple and complex searches to retrieve data.
`ldapmodify`	The ldapmodify Tool	Modify the attribute values of one or more entries, or add new entries.
`ldapdelete`	The ldapdelete Tool	Delete one or more entries given by their DN (distinguished name).
`ldapcompare`	The ldapcompare Tool	Compare attribute values with those given on the command-line.
`ldapcmp`	The ldapcmp Tool	Compare DNs and attribute values in entire subtrees of two directories.
`LDAPSubtdel`	The LDAPSubtdel Tool	Delete an entire subtree in a directory.
`DsmlSearch`	Directory Access Tools Using DSML	Same as `ldapsearch` but returns results in XML format.
`DsmlModify`	Directory Access Tools Using DSML	Same as `ldapmodify` but takes input in XML format.

Performance Evaluation

The performance evaluation tools help you run tests to measure your server's average response time to client requests. These tools perform repeated LDAP authentication, search, add, and delete operations to simulate actual usage. Use these tools before and after reconfiguring your directory to optimize performance. Run them regularly to monitor server response as directory size and usage evolves.

Table 1-2 Performance Evaluation Tools

Tool	Chapter	Purpose
`idsktune`	The idsktune Optimization Tool	Optimize your operating system and network settings for Sun ONE Directory Server.

Table 1-2 Performance Evaluation Tools *(Continued)*

Tool	Chapter	Purpose
`ldclt`	The ldclt Stress Test Tool	A powerful and flexible LDAP client tool for testing directory servers.
`rsearch`	The rsearch Search Tool	Measure the performance of search, compare, and delete operations.

LDAP v3 Tools

Lightweight Directory Access Protocol (LDAP) v3 specifies standards that provide both read and update access to a directory server.

Table 1-3 LDAP v3 Tools

Tool	Chapter	Purpose
`srchrate`	`The Search Performance Measurement Tool`	Measure search performance under high server loads.
`modrate`	`The Modify Performance Measurement Tool`	Measure the performance of modification operations.
`authrate`	`The Rate of Authentication Measurement Tool`	Measure the performance of connecting and performing authentication.
`infadd`	`The Add Performance Measurement Tool`	Measure the performance of add operations for creating new entries.

LDIF Deployment

LDIF (LDAP Data Interchange Format) is the standard format for importing and exporting directory contents. The LDIF deployment tools process large LDIF files, either generating, modifying, or comparing the LDAP entries and attribute values they contain. Use these tools to deploy a testing environment and generate large test databases, to make global directory updates off line, and to synchronize multiple unconnected databases.

Table 1-4 LDIF Deployment Tools

Tool	Chapter	Purpose
`create_instance.pl`	The Directory Server 4.x Instance Creation Tool	Use existing configuration files and interactive user input to generate an `.inf` file and create a new server instance (Directory Server 4.*x* only).
`dbgen.pl`	`The Standard Schema LDIF Generator Tool`	Generate random data for tests with the performance evaluation tools.
`ldifgen`	`The Custom Schema LDIF Generator Tool`	Generate random data for tests with legacy tools.
`MakeLDIF`	The Java-based LDIF Generator Utility	Another entry generator.
`ldifxform`	`The LDIF Transformation Tool`	Edit an LDIF file for global updates and extracts data for reports.
`mmldif`	`The LDIF Merge Tool`	Simulate a multi-master merge using LDIF files.
`ldiffer.pl`	`The LDAP Compare and Modify Tool`	Synchronizes differences between two directories.

Maintenance and Debugging

The maintenance and debugging tools help directory administrators to interpret logs and other trouble-shooting files. Use these tools to determine the causes of errors when they occur, as well as to perform preventive maintenance by monitoring directory usage and server statistics.

Table 1-5 Maintenance and Debugging Tools

Tool	Chapter	Purpose
`logconv.pl`	`The Log Analyzer Tool`	Interpret access logs and compile usages statistics.
`replcheck.pl`	The Replication Checker Tool	Verify whether two or more replicating servers are synchronized.
`migrateSchemaTo5.pl`	The Schema Migration Tool	Automate the process of updating your Sun ONE Directory Server 5.2 schema from 4.*x*.
`searchplay`	`The Search Operations Audit Tool`	Replay search operations found in the directory access log.
`viewldbg`	`The Core File Analyzer Tool`	Debug a core file (Solaris platforms only).

Table 1-5 Maintenance and Debugging Tools *(Continued)*

Tool	Chapter	Purpose
dbscan	The Database File Analyzer Tool	Create text output of Sun ONE Directory Server database files.
Open-source security tools	Network Security Services	Manage and debug security mechanisms used in client applications.
Unsuported Perl utilities	Unsupported Utilities	A set of Perl scripts provided in the *installDir*/unsupported/perl directory that provide examples of automated user and directory administration.

Sample Phonebook Application

With the JSP™ Directory Gateway (jdgw), you can design a web client to present directory contents in any browser. It consists of JavaServer™ Pages and an LDAP tag library. The JSP (which uses servlets to provide a web interface to access a directory server) use the LDAP tag library to write LDAP client servlets which then access a directory and generate the response in HTML. The sample application described in Chapter 32, "JSP Directory Gateway Phonebook" is a phone book called LookMeUp that searches for employee names in a corporate directory. Chapter 33, "Tag Library Reference" describes the tag libraries that can be used with the application.

NameFinder Application

NameFinder is a web-based tool to look up people in an LDAP database. The DSRK includes the web archive and other files related to deploying this application. Chapter 34, "NameFinder Application" describes this application.

Java Naming and Directory Interface

Java™ Naming and Directory Interface (JNDI) is an API used to provide naming and directory functionality to applications written in the Java programming language. Using JNDI, Java applications can store and retrieve Java objects of any type. It provides methods for performing standard directory operations, such as

associating attributes with objects and searching for objects using their attributes. The DSRK includes the Early Access 1 release of the JNDI DSML v2 Service Providers and the 1.0 release of the JNDI LDAP Booster Pack. Chapter 35, "Java Naming and Directory Interface" describes these tools.

Sun ONE Directory Server Plug-In

Chapter 36, "Attribute Value Uniqueness Plug-In" describes the Attribute Value Uniqueness plug-in which enforces the uniqueness of attribute values in a multi-master replication topology. This is an unsupported plug-in for Sun ONE Directory Server 5.2 only.

Sun ONE LDAP SDK for C

The DSRK bundles version 5.11 of the Sun ONE LDAP SDK for C. Use this library to write C or C++ client applications that take full advantage of the performance of the Sun ONE Directory Server. The API also includes extensions that give access to the latest features in Directory Server 5.2.

NOTE Because they are built around the core functions of the LDAP v2 and v3 standards, the API can be used to interact with any conforming LDAP server. This API conforms to IETF standard "LDAP Application Programming Interface," defined by RFC 1823 and now revised by `draft-ietf-ldapext-ldap-c-api-05`.

The API is defined by the header files that declare all of the functions, data types and code values that are available in the binaries. The complete API is documented in the *Sun ONE LDAP SDK for C Programming Guide*. The SDK also includes sample code that demonstrates how to call most of the functions.

NOTE The Sun ONE LDAP SDK for C is a binary release of the open source LDAP SDK for C source code available through `www.mozilla.org`. Updated releases are also available at:

`http://sunonedev.sun.com/`

Sun ONE LDAP SDK for Java

The DSRK bundles version 4.15 of the Sun ONE LDAP SDK for Java. The SDK consists of binary jar files containing all packages, classes and methods of the API. Equivalent in functionality to the SDK for C, Java client applications use this API to interact with LDAP directories. Whereas the Java Naming and Directory Interface™ (JNDI) provides a protocol-independent abstraction of directory services, this API exposes the LDAP-specific operations for direct access to an LDAP directory server. Use the classes and methods of the API to develop LDAP-enabled applets or applications for the J2EE™ platform or any of the Java platforms.

NOTE The Sun ONE LDAP SDK for Java conforms to the IETF standard "Java LDAP Application Program Interface," defined by `draft-ietf-ldapext-ldap-java- api-15`.

The Java API is documented in the *Sun ONE LDAP SDK for Java Programming Guide*. However, the Programming Guide does not include all of the latest updates to the API. Please refer to the corresponding Javadoc™ pages for the latest reference information.

NOTE The source code is also available as open source through `www.mozilla.org`, and updated releases are available at:

`http://sunonedev.sun.com/`

Chapter 2

Installing the Directory Server Resource Kit

This chapter provides instructions for installing Sun™ ONE Directory Server Resource Kit . It contains the following sections:

- Before Installing
- Solaris Operating System (SPARC and x86 Platform Editions)
- Windows® Operating Systems
- Additional Platforms
- Files and Directories

Before Installing

The Directory Server Resource Kit (DSRK) is available from Sun Microsystems' Download Center. To download it, type (or paste) the following URL into a browser window:

```
http://wwws.sun.com/software/download/products/3eef9a29.html
```

From this web page, click the Download link regardless of your platform. Login as noted or register with the Sun Download Center. Once logged in, read and accept the software license agreement. Accepting this agreement brings you to the page from which the platform-specific DSRK installation file can be downloaded.

NOTE To obtain a copy of the Directory Server Resource Kit on CD-ROM, speak with your Sun sales associate.

The DSRK can be installed on computers running the following operating systems:

- Solaris™ Operating System (SPARC® and x86 Platform Editions)
- Windows Operating Systems
- Additional Platforms: IBM AIX, HP-UX, and Red Hat Enterprise Linux

NOTE The Download Center contains two downloadable versions of the DSRK for each platform: debug and optimize. The debug version is a binary built with symbolic debug information. This version is used to develop and debug applications written with the DSRK. The optimized version does not contain the debug information. The two builds execute identically but because the debug build contains extra code, they differ in size.

Solaris Operating System (SPARC and x86 Platform Editions)

The DSRK is supported by the Solaris 8 and 9 operating systems on both SPARC and x86 machines. Although there is a separate installation package that contains native binaries for the designated version on each architecture , the package structure and the installation procedure is identical.

System Requirements

The DSRK requires 56 MB of hard disk space and an extra 37 MB of temporary space to install from the CD-ROM. Twice as much temporary space is needed if you download the zip file from the Download Center.

CAUTION Before beginning the installation, remove any previous versions of the DSRK and/or LDAP SDKs from your machine. Follow the uninstall instructions for the version of the software installed on your system.

Installing DSRK on Solaris

1. Locate the zipped DSRK file for your version of the Solaris operating system on the Download Center, and save it to an empty directory. The available files are:
    - Solaris Operating System 8 & 9 for the SPARC Platform

```
dsrk52-SunOS5.8_DBG.zip, 130.75 MB (debug)

dsrk52-SunOS5.8_OPT.zip, 55.55 MB (optimize)
```

- Solaris Operating System 9 for the x86 Platform

```
dsrk52-SunOS5.8_i86pc_DBG.zip, 117.12 MB (debug)

dsrk52-SunOS5.8_i86pc_OPT.zip, 47.98 MB (optimize)
```

Alternately, locate the corresponding file on the DSRK CD-ROM and copy it to an empty directory.

2. Extract the contents of the zip file into an empty directory with the following command:

   `$ unzip` *nameofDSRKfile.zip* `-d` *tempDir*

   tempDir is an empty directory used only during installation. The `-d` option may be omitted if the zip file has been downloaded to *tempDir*. The extracted files contain the DSRK install program, installation configuration files, and the software in compressed format.

3. Login as root.

4. Change to the directory in which the zipped files were extracted and run the installation program with the following command:

   ```
   # java DSRK
   ```

   The install program will lead you through the installation of the software. When asked for input, press Return to accept the default, or type a new value. You may also type Control-B to go back to the previous screen or Control-C to quit the program without installing.

5. Type Yes or yes to accept the software license.

   The default is to not accept the license and exit the install program.

6. Choose a location for the software to be installed.

   The default directory is the root directory. Some functionality of the DSRK tools relies on this location. You may have to specify additional tool options or set your `LD_LIBRARY_PATH` if you install the software in another location.

7. Press Return to install.

   The software is bundled as a single component to be installed all at once.

8. Use the following command to remove the temporary directory and all files it contains:

```
# rm -Rf tempDir/*
```

After the installation is complete, the temporary directory and its contents are no longer needed.

Once installed, the tools and APIs of the DSRK are ready to be used. See "Files and Directories" on page 54 for a description of the product contents.

Removing DSRK From Solaris

The product includes a program to remove the DSRK software from your machine. Use this application before installing an upgraded version of the product.

CAUTION If you customized any files in the software bundle and wish to keep them, you must rename them or copy them to another location outside of the installation directory. The uninstallation program removes all product files by name; it will not remove other files found under the installation directory. Any personally configured files may thus be kept and used with an upgraded version of the software when it is installed in the same location.

As root, change to the directory where the DSRK was installed and run the uninstall program:

```
# java uninstall_S1DSRK
```

The uninstall program will ask you to specify the components and subcomponents to remove. The software is a single component and will be removed all at once. Press Return twice to select the single component and remove all DSRK software.

NOTE If there are no custom files to keep (as detailed in the previous Caution note), the uninstall program will also remove the installation directory.

Windows® Operating Systems

The DSRK is supported on machines that run the Microsoft® Windows 2000 Advanced Server SP2 or the Windows NT® 4.0 SP6a operating systems.

System Requirements

The DSRK requires 37 MB of hard disk space and an extra 30 MB of temporary space to install from the CD-ROM. Twice as much temporary space is needed if you download the zip file from the Download Center.

Installing DSRK on Windows

1. Log in to Windows as a user with Administrator privileges.
2. Locate the zipped DSRK file for the Windows operating system from the Download Center, and save it to an empty folder. The available files are:

   `dsrk52-WINNT4.0_DBG.zip, 54.25 MB (debug)`

   `dsrk52-WINNT4.0_OPT.zip, 44.62 MB (optimize)`

   Alternately, locate the corresponding file on the DSRK CD-ROM and copy it to an empty folder.
3. Extract all contents of the zip file into the empty folder using a zip extraction utility.

   The empty folder is used only during installation. The extracted files will contain the DSRK install program, installation configuration files, and the software in compressed format.
4. Open the Command Prompt window by selecting Start > Programs > Accessories > Command Prompt.

   The DSRK install program must be run from the command-line.
5. Change to the directory in which the DSRK bits were extracted and run the following command:

   `#java DSRK`

   The install program will lead you through the installation of the software. When asked for input, press Return to accept the default, or type a new value. You may use the Back and Next buttons to modify any choices or the Cancel button to quit without installing.
6. Click Yes to accept the software license and continue installation.

7. Choose a location for the software to be installed.

   The default directory is the root directory. Some functionality of the DSRK tools relies on this location. You may have to specify additional tool options if you install the software in another location.

8. Click Install to install the software component.

   The install program automatically modifies the Windows Registry.

9. Delete the folder to which the zip file was extracted.

   After the installation is complete, the temporary directory and its contents are no longer needed.

Once installed, the tools and APIs of the DSRK are ready to be used. See "Files and Directories" on page 54 for a description of the product contents.

Post-Installation for Windows

The following steps should be taken after the installation on a machine running the Windows operating system.

Changing the Path Variable

The DSRK tools are designed to be launched from the Command Prompt, not from the Windows desktop. You should add the following library locations to the PATH environment variable of your command session:

```
PATH=%PATH%;installDir\lib;installDir\lib\nss\lib
```

Language Interpreters

Several tools in the DSRK require the Perl interpreter, version 5.005_03 or later. The Perl interpreter needs to be downloaded if not already installed. Refer to "Perl Development" on page 32 for links to the download.

Removing DSRK From Windows

The product includes an uninstall program to remove the DSRK software from your machine. Use this application before installing an upgraded version of the product.

CAUTION If you customized any files in the software bundle and wish to keep them, you must rename them or copy them to another location outside of the installation directory. The uninstallation program removes all product files by name; it will not remove other files found under the installation directory. Any personally configured files may thus be kept and used with an upgraded version of the software when it is installed in the same location.

1. Log in to Windows as a user with Administrator privileges.
2. Open the Command Prompt window by selecting Start > Programs > Accessories > Command Prompt.

   The DSRK uninstall program must be run from the command-line.
3. Change to the directory in which the DSRK bits were extracted and run the following command:

   ```
   #java uninstall_S1DSRK
   ```

   The uninstall program will ask you to specify the components and subcomponents to remove. The software is a single component and will be removed all at once. Press Return twice to select the single component and remove all DSRK software.

NOTE If there are no custom files to keep (as detailed in the previous Caution note), the uninstallation program will also remove the installation directory.

Additional Platforms

The DSRK is also supported on machines that run on the IBM-AIX, the HP-UX and the Red Hat Linux operating systems. Table 2-1 lists these operating systems and their disk space requirements.

Table 2-1 Additional Operating Systems For The DSRK

Platform	Disk Space Requirements For Installation
IBM AIX 4.33 or later	The product requires 54 MB of hard disk space and an extra 37 MB of temporary space during the installation procedure.
HP-UX B 11.0	The product requires 42 MB of hard disk space and an extra 33 MB of temporary space during the installation procedure.

Table 2-1 Additional Operating Systems For The DSRK

Platform	Disk Space Requirements For Installation
Red Hat Linux 7.2 on the Intel architecture	The product requires 50 MB of hard disk space and an extra 36 MB of temporary space during the installation procedure.

The temporary space requirements listed in Table 2-1 are for installation from a CD-ROM. Twice as much temporary space is needed if you download the zip file from the Download Center.

NOTE The following software installation and removal procedures may need to be tailored to the command environment available on your platform.

Installing DSRK on Additional Platforms

1. From the Download Center, locate the zipped DSRK file for the version of the operating system running on your machine and save it to an empty directory. The available files are:

   - IBM AIX 5.1 Platform

     ```
     dsrk52-AIX5.1_DBG.zip, 81.50 MB (debug)
     dsrk52-AIX5.1_OPT.zip, 60.88 MB (optimize)
     ```

   - HP-UX 11.11 Platform

     ```
     dsrk52-HP-UXB.11.11_DBG.zip, 115.36 MB (debug)
     dsrk52-HP-UXB.11.11_OPT.zip, 58.31 MB (optimize)
     ```

   - RedHat Linux 7.2 or Sun Linux 5.0 Platform

     ```
     dsrk52-Linux2.4_x86_DBG.zip, 74.02 MB (debug)
     dsrk52-Linux2.4_x86_OPT.zip, 48.27 MB (optimize)
     ```

   Alternately, locate the corresponding file on the DSRK CD-ROM and copy it to an empty directory.

2. Extract the contents of the zip file into an empty directory with the following command:

```
$ unzip dsrk52-platform.zip -d tempDir
```

tempDir is an empty directory used only during installation. The `-d` option may be omitted if the zip file has been downloaded to *tempDir*. The extracted files contain the DSRK install program, the installation configuration files, and the software in compressed format.

3. Login as root.
4. Change to the directory in which the zipped files were extracted and run the installation program:

   ```
   # java DSRK
   ```

   The install program will lead you through the installation of the software. When asked for input, press Return to accept the default, or type a new value. You may also type Control-B to go back to the previous screen or Control-C to quit the program without installing.
5. Type Yes or yes to accept the software license.

   The default is to not accept the license and exit the installation program.
6. Choose a location for the software to be installed.

   The default directory is the root directory. Some functionality of the DSRK tools relies on this location. You may have to specify additional tool options or set your `LD_LIBRARY_PATH` if you install the software in another location.
7. Press Return to install.

   The software is bundled as a single component to be installed all at once.
8. Use the following command to remove the temporary directory and all files it contains:

   ```
   $ rm -Rf tempDir/*
   ```

   After the installation is complete, the temporary directory and its contents are no longer needed.

Once installed, the tools and APIs of the DSRK are ready to be used. See “Files and Directories” on page 54 for a description of the product contents.

Removing DSRK From Additional Platforms

The product includes an uninstall program to remove the DSRK software from your machine. Use this application before installing an upgraded version of the product.

CAUTION If you customized any files in the software bundle and wish to keep them, you must rename them or copy them to another location outside of the installation directory. The uninstallation program removes all product files by name; it will not remove other files found under the installation directory. Any personally configured files may thus be kept and used with an upgraded version of the software when it is installed in the same location.

As root, change to the directory where the DSRK was installed and run the uninstall program:

```
# java uninstall_S1DSRK
```

The uninstall program will ask you to specify the components and subcomponents to remove. The software is a single component and will be removed all at once. Press Return twice to select the single component and remove all DSRK software.

NOTE If there are no custom files to keep (as detailed in the previous Caution note), the uninstallation program will also remove the installation directory.

Files and Directories

This guide uses *DSRK_base* as a variable place holder for the home directory where you have installed Directory Server Resource Kit. This directory contains the files and directories detailed in Table 2-2.

Table 2-2 Directories of the DSRK Installation

Directory	Contents
`NameFinder`	Contains elements for NameFinder, a web-based tool to search for people in an LDAP database.
`bin/dsrk52`	Contains executable binary files for most tools in the DSRK.
`data`	Location of the files for the database generator tool. See Chapter 18, “The Standard Schema LDIF Generator Tool.”
`examples/ldclt`	Contains examples for the LDAP client tool. See Chapter 11, “The ldclt Stress Test Tool.”
`java/LDAPSubtDel`	Contains components to install and run the LDAP tool used to delete subtrees. See Chapter 8, “The LDAPSubtdel Tool.”
`java/MakeLDIF`	Contains components to install and run the MakeLDIF tool.

Table 2-2 Directories of the DSRK Installation *(Continued)*

Directory	Contents
`java/DSMLtools`	Contains components to install an run DSML tools that use the SOAP/DSML interface to communicate with Directory Server. See Chapter 9, “Directory Access Tools Using DSML.”
`jdgw/apache-downloads`	Contains application server downloads for use with the JSP Directory Gateway phone book application. See Chapter 32, “JSP Directory Gateway Phonebook.”
`jdgw/ldap-taglib`	Contains the LDAP tag library for use with the JSP Directory Gateway phone book application. See Chapter 32, “JSP Directory Gateway Phonebook.”
`jdgw/phonebook-app`	Contains the JSP Directory Gateway phone book application. See Chapter 32, “JSP Directory Gateway Phonebook.”
`jndi-dsml`	Contains a zip archive of the Java Naming and Directory Interface (JNDI) LDAP Booster Pack 1.0.
`jndi-ldap`	Contains a zip archive of the JNDI DSML v.1 Service Provider, 1.2.
`lib/`	Contains binary libraries needed for running the tools, including the same binaries as contained in the Sun ONE LDAP SDK for C.
`lib/ldapcsdk`	The base directory for the Sun ONE LDAP SDK for C. Its subdirectories contain the include files and the binary libraries needed to develop and run applications using this C API.
`lib/ldapjdk`	The base directory for the Sun ONE LDAP SDK for Java. It contains the jar files needed to develop and run applications using this Java API.
`lib/nss`	The base directory for the Netscape Security Services, containing subdirectories for the run-time libraries and tools binaries.
`lib/tcl8.2`	Contains Tcl (Tool Command Language) scripts. Tcl is an open source scripting language. This directory contains a complete Tcl interpreter.
`perl`	Contains Perl scripts for the Perl-based tools.
`unsupported`	Contains sample Perl scripts and the PerLDAP tool, all provided “as-is” with no support and no endorsement.

Part 2

Directory Access Tools

Chapter 3

The ldapsearch Tool

The ldapsearch tool issues search requests to an Lightweight Directory Access Protocol (LDAP) directory and displays the result as LDAP Data Interchange Format (LDIF) text. Its many options allow you to perform different types of search operations, from simple entry retrieval to advanced searches that involve security or directory referrals. This chapter provides instructions on how to use the ldapsearch tool. It contains the following sections:

- Overview
- Command Usage
- Return Values
- Command-Line Examples

Overview

ldapsearch is a command-line tool that opens a connection to an LDAP server, binds to it, and performs a search using a filter. The results are then displayed in the LDIF.

NOTE The LDIF is used to represent LDAP entries in a simple text format. See Appendix E, "LDAP Data Interchange Format," in the *Sun ONE Directory Server Reference Manual* for more information.

The ldapsearch tool is also provided with Sun™ ONE Directory Server in the *DirectoryServer_base*/shared/bin directory. However, the DSRK and its updates should include the latest version of the tool in the *DSRK_base*/bin/dsrk52 directory.

CAUTION If you use the Solaris™ operating environment, there may be an older version of `ldapsearch` in `/usr/bin`. Be sure your path is set to use the latest version in *DSRK_base*`/bin/dsrk52`.

Command Usage

A search involves binding and possibly authenticating to a directory server before initiating a search operation with a certain scope from a given base DN. The request includes a filter of the attribute values that must match in the entries returned. The command-line options allow you to sort the results, limit how much information is returned, control how referrals are followed, enable a secure connection, and set a time limit for the operation. Results of the search are displayed as LDIF text to the standard output. By default, the results contain the DN and all attributes for each entry found by the search. The results may also be reformatted using command-line options.

Syntax

The syntax of the `ldapsearch` tool on the command-line can take any of these forms:

```
ldapsearch -b "baseDN" [ options ]  "filter"  [ attributeName ... ]
ldapsearch -b "baseDN" [ options ]  -f filterFile  [ attributeName ...]
```

Where:

- *baseDN* is the base of the search, usually enclosed in double quotes ("") for the shell. The `-b` *baseDN* parameter may be omitted if the `LDAP_BASEDN` environment variable is set.
- *options* are the command-line options and their parameters described in “Options” on page 61.
- *filter* is an RFC 2254-compliant LDAP search filter, usually in double quotes ("") for the shell. “LDAP Search Filters” in Chapter 4 of the *Sun ONE Directory Server Getting Started Guide* details how to configure a filter.
- The *filterFile* contains one LDAP search filter per line, each one being used for a separate search. A command-line filter cannot be specified when using the `-f` option.

NOTE In the first syntax, a filter is configured on the command-line and, in the second, the filter is defined in a separate file.

- One or more *attributeNames* specifies the list of attributes and corresponding values to be returned for each entry that matches the filter. When the list of attributes is omitted, `ldapsearch` returns all attributes permitted by the access rights of the bind DN, with the exception of operational attributes.

NOTE To retrieve operational attributes, you must explicitly specify their *attributeName*. To retrieve all regular attributes in addition to operational attributes, append an asterisk (*) to the attribute list.

Options

The `ldapsearch` tool has four types of options:

- Common Options
- Input And Output Options
- LDAP Controls Options
- SSL (Secure Socket Layer) Options

The following sections detail these options. The `ldapsearch -H` command and option when run on the command-line will display brief descriptions of all the command-line options.

Common Options

The common options listed in Table 3-1 control the binding and general behavior of the `ldapsearch` command.

Table 3-1 Common Options for `ldapsearch`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. When this option is omitted, the default is `localhost`.
`-p`	*port*	Specify the port number for accessing the directory server host. The default is 389 normally and 636 when the SSL options are used.

Table 3-1 Common Options for `ldapsearch` *(Continued)*

Option	Parameter	Purpose
`-D`	*bindDN*	Specify a bind DN for accessing your directory, usually in double quotes ("") for the shell. If the bind DN and its password are omitted, the tool will use anonymous binding. The bind DN determines what entries and attributes will appear in the search results, according to the DN's access permissions.
`-w`	*password*	Specify the password for the bind DN. **CAUTION: Specifying the password on the command-line is a possible security risk.**
`-w`	`-`	Type the password for the bind DN when prompted. This is the most secure way of specifying the password.
`-j`	*filename*	Specify a file containing the password for the bind DN. Use this option in scripts and place the password in a secure file to protect it. This option is mutually exclusive with the `-w` option.
`-b`	*baseDN*	Specify the base DN for the search operation, usually in double quotes ("") for the shell. You may omit this option if you specify the base DN in the `LDAP_BASEDN` environment variable.
`-s`	*scope*	Specify the scope of a search. The *scope* parameter may have one of the following values: • `base` - For searching only the base entry. • `one` - For searching only the children of the base entry. • `sub` - For searching the base entry and all its descendants. This is the default if the `-s` option is omitted.
`-f`	*filterFile*	Specify the name of a file containing filter strings. This file contains one or more filters, each on a separate line; `ldapsearch` will perform a separate search with each filter, in the order found in the file.
`-l`	*seconds*	Specify 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, whose default is 3,600 seconds. For more information, see "nsslapd-timelimit (Time Limit)" in Chapter 4 of the *Sun ONE Directory Server Reference Manual*.
`-V`	*version*	Specify the LDAP protocol version number to be used for the search operation, either 2 or 3. LDAP v3 is the default; only specify LDAP v2 when connecting to servers that do not support v3.
`-Y`	*proxyDN*	Specify the proxy DN to use for the search operation, usually in double quotes ("") for the shell. For more information about proxy authorization, see Chapter 6, "Managing Access Control," in the *Sun ONE Directory Server Administration Guide*.

Table 3-1 Common Options for `ldapsearch` *(Continued)*

Option	Parameter	Purpose
`-a`	*aliasMode*	Specify how aliases are dereferenced when encountered in a search. Note that Sun ONE Directory Server 5.2 and previous versions do not support aliases, so this option has no effect on these servers. The parameter may be one of the following values: • `never` - Aliases are never dereferenced; this is the default. • `find` - Aliases are dereferenced only while finding the base DN. • `search` - Aliases are dereferenced when searching entries below the base DN (but not when finding the base DN). • `always` - Aliases are dereferenced both when finding the base DN and searching beneath it.
`-M`		Manage smart referrals: when referrals are part of the search results, return the actual entry containing the referral instead of the entry obtained by following the referral. See “Creating Smart Referrals” in Chapter 2 of the *Sun ONE Directory Server Administration Guide*.
`-O`	*hopLimit*	(Capital letter O) Specify the maximum number of referral hops to follow while searching.
`-R`		Specify that referrals should *not* be followed. By default, referrals are followed automatically.
`-v`		Verbose output mode: the tool will display additional information about the search, such as the filter string and the number of results for each search.
`-n`		No-op mode: use with the `-v` option to show what the tool would do with the given input but do not actually perform the search.
`-0` (zero)		Allow runtime library version mismatches. When this option is omitted, the default behavior is to assert that the revision number of the LDAP API is greater than or equal to that used to compile the tool. Also, if the API library and the tool have the same vendor name, the tool will also assert that the vendor version number of the API is greater than or equal to that used to compile the tool. This information is based on the contents of the `LDAPAPIInfo` structure. (See the *Sun ONE LDAP SDK for C Programming Guide*.)
`-H`		Display the usage help text that briefly describes all options.

Input And Output Options

The input and output options listed in Table 3-2 control how the `ldapsearch` results are sorted and presented.

Table 3-2 Input and Output Options for `ldapsearch`

Option	Parameter	Purpose
`-i`	*locale*	Specify the character set to use for command-line input. The default is the character set specified in the `LANG` environment variable. This option can be used to convert from the specified character set to UTF8, thus overriding the `LANG` setting. You can also input a specific character set for the bind DN, base DN, and the search filter pattern. The `ldapsearch` tool converts the input from these arguments before it processes the search request. For example, `-i no` indicates that the bind DN, base DN, and search filter are provided in Norwegian. This argument only affects the command-line input. A filter file (specified with the `-f` option) will not be converted by `ldapsearch`.
`-k`	*path*	Specify the path to a directory containing conversion routines. Use these routines if you wish to specify a sorting language that is not supported by your directory server. See Appendix C, “Directory Internationalization” in the *Sun ONE Directory Server Reference Manual*.
`-S`	[`-`]*attribute*	Specify an attribute to sort entries returned by the search. The sort criteria is alphabetical based on the attribute’s value or reverse alphabetical with the form `-`*attribute*. You may give multiple `-S` options to refine the sorting, for example: `-S sn -S givenname`. By default, the entries are not sorted. Use the `-x` option described in Table 3-3 to perform server-side sorting.
`-z`	*max*	Specify the maximum number of entries to return in response to a search request. Regardless of the value specified here, `ldapsearch` will never return more entries than the number allowed by the directory server’s `nsslapd-sizelimit` attribute whose default is 2,000 entries. See “nsslapd-sizelimit (Size Limit)” in Chapter 4 of the *Sun ONE Directory Server Reference Manual*. This limitation does not apply if you bind as the root DN (with `-D "cn=directory manager"`), in which case this option defaults to `0` (zero) and the size limit attribute is overridden.
`-A`		Specify that the search retrieve only attribute names, not the attribute values. This option is useful if you just want to determine if an attribute is present for an entry.
`-1`		Omit the leading `"version: 1"` line in the LDIF output.
`-u`		User-friendly DNs: specify that this alternate form of the DN also be included in the output of a search result entry, in addition to the complete form that is always displayed.
`-t`		Temporary file output: each attribute of each entry in the search results will be written to a separate file in the system’s temporary directory (usually `/tmp`). The standard output of the tool will include the name of the file instead of the attribute’s value. When using this option, no base-64 encoding is performed on the values, regardless of content.

Table 3-2 Input and Output Options for `ldapsearch` *(Continued)*

Option	Parameter	Purpose
`-U`		URL format : when using temporary file output, the standard output of the tool will include the URL of the file instead of the attribute's value, for example: `jpegPhoto:< file:/tmp/ldapsearch-jpegPhoto-YzaOMh`. **NOTE: This option is valid only with the `-t` option.**
`-T`		Format the output of search results so that no line breaks are used within individual attribute values.
`-o`		Used to specify Simple Authentication and Security Layer (SASL) options (mech, realm, authid and authzid). For more information on these options, see "Configuring LDAP Clients to Use Security" in Chapter 11 of the Sun ONE Directory Server Administration Guide.
`-F`	*separator*	Format the output of search results so that the given *separator* is used between attribute names and their values. **NOTE: This option may be used only in conjunction with the `-o` option.**
`-B`		Format the output of search results to print binary values as they are stored in the directory. When used in conjunction with `-o`, the binary data in the output will not use base-64 encoding.

LDAP Controls Options

The options in Table 3-3 provide advanced search controls for server-side sorting, virtual lists, and persistent searches. This functionality is available only if the server supports the corresponding LDAP controls. These options will also display any additional information that the server sends in response to the control.

Table 3-3 LDAP Controls Options for `ldapsearch`

Option	Parameter	Purpose
`-x`		Use with the `-S` option (in Table 3-2) to specify that search results be sorted on the server rather than by the `ldapsearch` command running on the client. This is useful if you want to sort according to a matching rule, as with an international search. It is usually faster to sort on the server, if supported, rather than on the client.

Table 3-3 LDAP Controls Options for `ldapsearch` *(Continued)*

Option	Parameter	Purpose
`-C`	*pattern*	Used to perform a search that keeps the connection open and displays results whenever entries matching the scope and filter of the search are added, modified, or removed. For this persistent search, the `ldapsearch` tool will run indefinitely. Control-C must be typed to stop it. By default, the tool will instruct the server to return entry change controls with the persistent search results. These controls indicate the type of operation that caused the entry to be detected by the search. *pattern* has the format: `ps:`*changeType*`[:`*changesOnly*`[:`*entryChangeControls*`]]` *changeType* determines which modifications to an entry are detected and displayed in the output; its possible values are `add`, `delete`, `modify`, `moddn`, or `any`. *changesOnly* is an optional boolean value. The default `1` displays changes when they occur. Specify `0`, `f`, or `false` to display the results of the search before waiting for changes. *entryChangeControls* is also an optional boolean value. Specify `0`, `f`, or `false` if you do not want the server to return entry change controls. In this case, you must also specify a value for the *changesOnly* parameter.
`-G`	*pattern*	This virtual list view retrieves only a portion of all results, as determined by the index or value of the search target and the number of entries to be returned before and after the target. This option always requires the `-S` and `-x` options to specify the sorting order on the server. The pattern has two possible formats: • *entriesBefore:entriesAfter:value* - Specify the search target as the first entry in the sorted results for which the sort attribute is "greater than" or equal to the given *value*. For example, `-S sn -x -G 5:10:johnson` will return 16 entries in alphabetical order of the surname attribute: 5 less than `johnson`, the entry equal to or following `johnson`, and the 10 subsequent entries. • *entriesBefore:entriesAfter:index:count* - Specify the search target as the *index* position relative to the estimated *count*. If the count is `0` (zero), the index is taken as the absolute index of the target entry within the actual number of entries found. An *index* of `1` will always select the first entry in the sorted list of results. Otherwise, the target index is the first entry in slice of the list represented by the fraction *index/count*. For example, `-G 5:10:2:4` specifies the index closest to the beginning of the second quarter of the entire list. If the search yielded 100 entries, the target index would be 26, and this pattern would return entries 21 through 36. Give an index greater than the count to specify the last search result in the list. The number of *entriesBefore* and *entriesAfter* displayed may be limited by the beginning and end of the virtual list. `ldapsearch` takes results and displays the control response to give the total count of entries in the virtual list and the actual index of the target entry. Use these values to refine the search with more accurate *index* and *count* parameters.

SSL (Secure Socket Layer) Options

The options in Table 3-4 allow you to use LDAPS (LDAP over SSL) to establish a secure connection for the search. These options are valid only when LDAPS has been enabled and configured in your SSL-enabled directory server. For information on certificate-based authentication and creating a certificate database for use with LDAP clients, see Chapter 11, "Implementing Security," in the *Sun ONE Directory Server Administration Guide*. See "Using Authentication" on page 72 for examples using the SSL options.

Table 3-4 SSL Options for `ldapsearch`

Option	Parameter	Purpose
`-P`	*path*	Specify the path and filename of the client's certificate database. This file may be the same as the certificate database for an SSL-enabled version of Netscape™ Communicator, if available; for example: `-P /home/`*uid*`/.netscape/cert7.db` . When using the command on the same host as the directory server, you may use the server's own certificate database, for example:`-P` *installDir*`/slapd-`*serverID*`/alias/cert7.db`. Use the `-P` option alone to specify server authentication only.
`-Z`		Specify that SSL be used to provide certificate-based client authentication. This option requires the `-N` and `-W` options and any other of the SSL options needed to identify the certificate and the key database.
`-N`	*certificate*	Specify the certificate name to use for certificate-based client authentication, for example: `-N "Directory-Cert"`.
`-m`	*path*	Specify the path to the security module database. For example: `/usr/iplanet/servers/slapd-`*serverID*`/secmodule.db` You need to specify this option only if the security module database is in a different directory from the certificate database itself.
`-K`	*keyFile*	Specify the file and path name of the client's private key database. This option may be omitted if the key database is in the location already given by the `-P` option.
`-W`	*password*	Specify the password for the client's key database given in the `-K` or `-P` options. This option is required for certificate-based client authentication.

Return Values

The `ldapsearch` tool is based on the Sun ONE LDAP SDK for C, and its return values are those of the functions it uses, such as `ldap_simple_bind_s()`, `ldap_search_ext()`, and `ldap_result()`. These functions return both client-side and server-side errors and codes. Table 3-5 shows the possible return values when the directory is hosted on Sun ONE Directory Server. Other LDAP servers may send these values under different circumstances or may send different values. They may also send other result codes entirely; for example, custom result codes from a custom plug-in. For further information about result codes, see the *Sun ONE LDAP SDK for C Programming Guide*.

Table 3-5 Return Values of `ldapsearch`

Return Value	Result Code and Explanation
0 (0x00)	`LDAP_SUCCESS`: the operation was successful.
1 (0x01)	`LDAP_OPERATIONS_ERROR`: sent by Directory Server for general errors encountered by the server when processing the request.
2 (0x02)	`LDAP_PROTOCOL_ERROR`: the search request did not comply with the LDAP protocol. Directory Server may send this error code if it could not sort the search results or could not send sorted results.
3 (0x03)	`LDAP_TIMELIMIT_EXCEEDED`: sent by Directory Server if the search exceeded the maximum time specified by the `-l` option.
4 (0x04)	`LDAP_SIZELIMIT_EXCEEDED`: sent by Directory Server if the search found more results than the maximum number of results specified by the `-z` option.
10 (0x0a)	`LDAP_REFERRAL`: sent by Directory Server if the given base DN is an entry not handled by the current server and if the referral URL identifies a different server to handle the entry.
11 (0x0b)	`LDAP_ADMINLIMIT_EXCEEDED`: sent by Directory Server if the search found more results than the limit specified by the `lookthroughlimit` directive in the `slapd.conf` configuration file. If not specified in the configuration file, the default limit is 5000.
21 (0x15)	`LDAP_INVALID_SYNTAX`: sent by Directory Server if your substring filter contains no value for comparison.
32 (0x20)	`LDAP_NO_SUCH_OBJECT`: sent by Directory Server if the given base DN does not exist and if no referral URLs are available.
50 (0x32)	`LDAP_INSUFFICIENT_ACCESS`: sent by Directory Server if the DN used for authentication does not have permission to read from the directory.
53 (0x35)	`LDAP_UNWILLING_TO_PERFORM`: sent by Directory Server if the database is read-only.

Table 3-5 Return Values of `ldapsearch` *(Continued)*

Return Value	Result Code and Explanation
81 (0x51)	`LDAP_SERVER_DOWN`: the LDAP server did not receive the request or the connection to the server was lost.
82 (0x52)	`LDAP_LOCAL_ERROR`: an error occurred when receiving the results from the server.
83 (0x53)	`LDAP_ENCODING_ERROR`: the request could not be BER-encoded.
84 (0x54)	`LDAP_DECODING_ERROR`: an error occurred when decoding the BER-encoded results from the server.
85 (0x55)	`LDAP_TIMEOUT`: the search exceeded the time specified by the `-l` option.
87 (0x57)	`LDAP_FILTER_ERROR`: an error occurred when parsing and BER-encoding a search filter specified on the command-line or in a filter file.
89 (0x59)	`LDAP_PARAM_ERROR`: one of the options or parameters is invalid.
90 (0x5a)	`LDAP_NO_MEMORY`: memory cannot be allocated as needed.
91 (0x5b)	`LDAP_CONNECT_ERROR`: the specified hostname or port is invalid.
92 (0x5c)	`LDAP_NOT_SUPPORTED`: the `-V 2` option is needed to access a server that only supports LDAP v2.

Command-Line Examples

The examples in this section demonstrate common uses of the `ldapsearch` tool to access a directory. All examples assume the following context:

- You want to perform a search of all entries in the directory.
- All entries in the directory are stored under `dc=example,dc=com`.
- The directory has been configured to support anonymous access for search and read. Therefore, you do not have to specify any bind information in order to perform the search.
- The server is located on a machine with the given *hostname*.
- The server uses port number 389. Because this is the default port, you do not have to specify the port number on the search request.
- SSL is enabled for the server on port 636 (the default SSL port number).

Returning All Entries

Given the context, the following command will return all entries in the directory:

```
$ ldapsearch -h hostname -b "dc=example,dc=com" -s sub "objectclass=*"
```

The "objectclass=*" parameter is a search filter that matches any entry in the directory. The scope is set to the full subtree of the base DN (-s sub), and no attribute list is given, so all attributes for all entries will be returned.

Narrowing a Search

To narrow a search, specify a search filter enclosed in quotation marks directly on the command-line. Then, you can ask to receive only those attributes that you need. For example:

```
$ ldapsearch -h hostname -b "dc=example,dc=com" "cn=babs jensen" mail
telephonenumber
```

In this example, the search will return only the mail and telephonenumber attributes of all entries with a common name (cn) that matches "babs jensen".

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 view this entry by performing a search with an empty search base (-b ""). You must also specify a search scope of base and a filter of "objectclass=*", as follows:

```
$ ldapsearch -h hostname -b "" -s base "objectclass=*"
```

Searching the Schema Entry

Sun ONE Directory Server stores all directory server schema in the special entry with the DN "cn=schema". 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 hostname -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 avoid specifying the search base with the -b option every time you use the ldapsearch tool.

NOTE For information on how to set environment variables, see the documentation for your operating environment.

Typically, you set LDAP_BASEDN to your directory's root suffix value. Because the root suffix is the topmost entry in your directory, all searches will be able to scan the entire directory tree. For example, suppose you have set LDAP_BASEDN to dc=example,dc=com. To search for cn=babs jensen in your directory, use the following command-line:

```
$ ldapsearch -h hostname "cn=babs jensen"
```

In this example, the default scope is sub because the -s option was not used to specify a scope other than the base DN.

Using a Filter File

You can store search filters in a file instead of entering them on the command-line. When creating a filter file, specify each search filter on a separate line. The ldapsearch command runs a separate search with each filter in the order in which they appear in the file. This example uses a file named myFilters reproduced in Code Example 3-1.

Code Example 3-1 myFilters Filter File

```
sn=Francis
givenname=Richard
```

Assuming the search base is defined by the LDAP_BASEDN environment variable, the following command returns all entries that match either search filter:

```
$ ldapsearch -h hostname -f myFilters
```

In the output, ldapsearch first displays all entries with the surname Francis, and then all entries with the given name Richard. The two searches are independent, so an entry that matches both search criteria will be returned twice.

You can limit the set of attributes returned 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 surname and the given name attributes of each entry:

```
$ ldapsearch -h hostname -f myFilters sn givenname
```

Specifying Commas in 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 "Office Bolivia, S.A." subtree of the directory, use the following command:

```
$ ldapsearch -h hostname \
             -b "o=Office Bolivia\, S.A.,dc=example,dc=com" \
              "objectclass=*"
```

Using Authentication

There are two levels of authentication that the directory server may enforce with clients such as the ldapsearch tool: server authentication and client authentication. In server authentication, the server accepts connections only from clients that have a trusted certificate. In the stronger client authentication the client must sign the certificate with a password-protected private key.

NOTE In both cases, use the -p option to specify the directory server's SSL port. All other non-SSL options retain their original meaning and may be used as necessary.

Using Server Authentication

To perform a search with server authentication, use only the -P SSL option [as discussed in "SSL (Secure Socket Layer) Options" on page 67] on the command-line, in addition to other common options.

```
$ ldapsearch -h hostname -p 636 -b "dc=siroe,dc=com" \
             -D "uid=bjensen,dc=example,dc=com" -w bindPassword \
             -P /home/bjensen/certs/cert.db \
              "givenname=Richard"
```

Using Client Authentication

To perform a search with client authentication, you must give all SSL options [as discussed in “SSL (Secure Socket Layer) Options” on page 67] on the command-line, in addition to other common options.

```
$ ldapsearch -h hostname -p 636 -b "dc=example,dc=com" \
             -Z -P /home/bjensen/security/cert.db -N "bjscert" \
             -K /home/bjensen/security/key.db -W KeyPassword \
              "givenname=Richard"
```

CAUTION Do not use the `-D` and `-w` common options with client authentication, as the bind operation will use the authentication credentials specified with `-D` and `-w` instead of the certificate credentials desired.

Chapter 4

The ldapmodify Tool

The `ldapmodify` tool edits the contents of a Lightweight Directory Access Protocol (LDAP) directory, either by adding new entries or modifying existing ones. This chapter provides instructions on how to use the `ldapmodify` tool. It contains the following sections:

- Overview
- Command Usage
- Return Values
- Command-Line Examples

Overview

The `ldapmodify` tool takes entry updates, defined using the LDAP Data Interchange Format (LDIF), as input and issues a corresponding LDAP request to the designated directory server. The LDIF information can be configured in a file or directly at the command-line.

TIP By placing all entry update statements in a file, `ldapmodify` can be used to process large numbers of modifications as well as transferring entries between directories.

`ldapmodify` is also provided with Sun™ ONE Directory Server in the *DirectoryServer_base*`/shared/bin` directory. However, the DSRK and its updates should include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory.

CAUTION If you use the Solaris™ operating environment, there may be an older version of `ldapmodify` in `/usr/bin`. Be sure your path is set to use the latest version in *DSRK_base*`/bin/dsrk52`.

Command Usage

The `ldapmodify` tool processes entry update statements, or *change records*, defined by the LDIF . A change record contains the DN (distinguished name) of the target entry, the operation to perform, and any data needed to complete the operation. Code Example 4-1 is a change record configured to change Barbara Jensen's surname [`sn`] attribute to Morris. In addition, it will change all values of the multi-valued `cn` attribute. (When using the replace syntax, all current values of the specified attribute will be removed and all given values will be added.)

Code Example 4-1 Entry Update Statement or Change Record

```
ldapmodify -h host -p port -D "cn=Directory Manager" -w password
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: sn
sn: Morris
-
replace: cn
cn: Barbara Morris
cn: Babs Morris
```

`ldapmodify` reads any number of these change records from the command-line or from a file, modifying the corresponding entries according to the LDIF instructions. For each DN in the LDIF file, the tool will perform the requested LDAP operation (defined by `changetype`) on the designated entry. `ldapmodify` supports the following operations:

- add an entry
- delete an entry
- edit an entry (including the DN or relative DN)

Syntax

The syntax of the `ldapmodify` tool on the command-line can take any of these forms:

```
ldapmodify [ options ]
ldapmodify [ options ] < LDIFfile
ldapmodify [ options ] -f LDIFfile
```

Where:

- *options* are the command-line options and their parameters described in "Options" on page 78.
- *LDIFfile* is an RFC 2849-compliant LDIF text file containing new entries or updates to existing entries.

In its first form (without *LDIFfile)*, `ldapmodify` takes one or more LDIF update statements configured at the command-line, and ends the input with an end-of-file (EOF) marker. Once you enter all update statements and the EOF marker, `ldapmodify` will process the input and perform all operations.

NOTE The EOF marker is platform dependent:

- Type Control-D (^d) on most UNIX systems.
- Type Control-Z (^z) and press Enter on Windows® NT®.

The next two syntaxes take an LDIF file as input. The second syntax uses < (the *less than* symbol) to take the input from the specified file instead of the keyboard. The final syntax does the same by using the `-f` option. Some samples of syntax and update statements are given in "Command-Line Examples" on page 84.

NOTE For general information on LDIF, see Appendix E, "LDAP Data Interchange Format," in the *Sun ONE Directory Server Reference Manual*). Additional information on LDIF and update statements is in "Managing Entries From the Command Line" in Chapter 2 of the *Sun ONE Directory Server Administration Guide.*

Modification Prerequisites

When modifying the contents of a directory, you must satisfy several prerequisite conditions. First, the bind DN and password used for authentication must have the appropriate permissions for the operations being performed. (Many high level directory operations, such as creating a database suffix, may only be performed by the Directory Manager with a bind DN of `"cn=directory manager"`.) Second, if schema checking is active in your directory, Directory Server will check the contents of new and modified entries against the object class definition in the LDAP schema. All attributes of an entry, even those not being modified, are checked against the schema and must meet the following conditions:

- The value and value type of all attributes being added or modified must conform to their definition in the entry's object class. When this is not the case, the modification of this entry will fail.
- Attributes and values not being modified must also conform to the schema. The modification of the entry will fail even if the offending attribute is not being modified. This situation can even occur if you run Directory Server with schema checking turned off, remove a required attribute or set an illegal value, and then turn schema checking on. For more information, see Chapter 9, "Extending the Directory Schema," in the *Sun ONE Directory Server Administration Guide.*

Finally, you must ensure the coherent placement of entries in the LDIF input. Updates are performed in the order they are given in the input, allowing you to manage dependencies between operations. For example, if you want to add entries to a subtree that doesn't exist, your LDIF input must first give the update statement for adding the subtree entry, before the update statements for adding entries under the subtree.

CAUTION When a modification fails, only the operation on the faulty entry is affected. `ldapmodify` will stop processing further input although all entries processed before the error was encountered will be successfully added or modified. Use the `-c` option to specify that the tool should continue processing.

Options

The `ldapmodify` tool has three types of options:

- Common Options
- Input And Output Options

- SSL (Secure Socket Layer) Options

The following sections detail these options. The `ldapmodify -H` command and option when run on the command-line will display text that briefly describes all of the command-line options.

Common Options

The common options listed in Table 4-1 control the binding and general behavior of the `ldapmodify` command.

Table 4-1 Common Options for `ldapmodify`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. When this option is omitted, the default is `localhost`.
`-p`	*port*	Specify the port number for accessing the directory server host. The default is 389 normally and 636 when the SSL options are used.
`-D`	*bindDN*	Specify a bind DN for accessing your directory, usually in double quotes ("") for the shell. If the bind DN and its password are omitted, the tool will use anonymous binding. The bind DN determines what entries and attributes may be modified, according to the DN's access permissions.
`-w`	*password*	Specify the password for the bind DN. **CAUTION: Specifying the password on the command-line is a possible security risk.**
`-w`	`-`	Type the password for the bind DN when prompted. This is the most secure way of specifying the password.
`-j`	*filename*	Specify a file containing the password for the bind DN. Use this option in scripts and place the password in a secure file to protect it. This option is mutually exclusive with the `-w` option.
`-f`	*LDIFfile*	Give the name of a file containing LDIF update statements or new entries. (See "Syntax" on page 77.) The tool will perform each of the update operations (add, modify, or delete) in the order given in the file. When this option is omitted, `ldapmodify` will read LDIF update statements from the standard input (command-line).
`-B`	*baseDN*	Specify the base DN when performing additions, usually in double quotes ("") for the shell. All entries will be placed under this suffix, thus providing bulk import functionality.
`-V`	*version*	Specify the LDAP protocol version number to be used for the modify operation, either 2 or 3. LDAP v3 is the default; only specify LDAP v2 when connecting to servers that do not support v3.

Table 4-1 Common Options for `ldapmodify` *(Continued)*

Option	Parameter	Purpose
`-Y`	*proxyDN*	Specify the proxy DN to use for the modify operation, usually in double quotes (`""`) for the shell. For more information about proxy authorization, see Chapter 6, “Managing Access Control,” in the *Sun ONE Directory Server Administration Guide*.
`-M`		Manage smart referrals: when they are the target of the update, modify the actual entry containing the referral instead of the entry obtained by following the referral. For more information, see “Creating Smart Referrals” in Chapter 2 of the *Sun ONE Directory Server Administration Guide*.
`-O`	*hopLimit*	(Capital letter O) Specify the maximum number of referral hops to follow while finding an entry to modify. By default, there is no limit.
`-R`		Specify that referrals should *not* be followed. By default, referrals are followed automatically.
`-q`		Quiet output mode: the tool will not display any output about the operations it performs.
`-v`		Verbose output mode: the tool will display additional information about the operations it performs.
`-n`		No-op mode: use with the `-v` option to show what the tool would do with the given input but do not perform any operations.
`-0` (zero)		Allow runtime library version mismatches. When this option is omitted, the default behavior is to assert that the revision number of the LDAP API is greater than or equal to that used to compile the tool. Also, if the API library and the tool have the same vendor name, the tool will also assert that the vendor version number of the API is greater than or equal to that used to compile the tool. This information is based on the contents of the `LDAPAPIInfo` structure. (See the *Sun ONE LDAP SDK for C Programming Guide*.)
`-H`		Display the usage help text that briefly describes all options.

Input And Output Options

The input and output options listed in Table 4-2 control how `ldapmodify` processes input files and handles errors.

Table 4-2 Input and Output Options for `ldapmodify`

Option	Parameter	Purpose
`-a`		The add entry mode provides an easy way to add entries in LDIF. All input entries that do not contain an LDIF `changetype` statement and keyword are processed as adds; entries with a defined `changetype` statement are processed accordingly. In particular, this option allows you to directly add entries from the output files of the `ldapsearch` tool.
`-F`		Force the application of all updates, regardless of the replica status.
`-i`	*locale*	Specify the character set to use for the `-f` *LDIFfile* or standard input. The default is the character set specified in the `LANG` environment variable. You might want to use this option to perform the conversion from the specified character set to UTF8, thus overriding the `LANG` setting.
`-k`	*path*	Specify the path to a directory containing conversion routines. These routines are used to specify a locale that is not supported by default by your directory server. For more information, see Appendix C, “Directory Internationalization” in the *Sun ONE Directory Server Reference Manual*.
`-b`		Handle binary files: the `ldapmodify` tool will scan every attribute value in the input to determine whether it is a valid file reference, and if so, it will use the contents of the file as the attribute’s value. This option is used to input binary data for an attribute, such as a JPEG image. For example, the corresponding LDIF input would be: • `jpegPhoto: /tmp/photo.jpg` (on a UNIX platform) • `jpegPhoto: c:\tmp\photo.jpg` (on Windows) The `ldapmodify` tool also supports the LDIF `: <` *URL* notation for directly including file contents. For example: • `jpegPhoto: < file:///tmp/photo.jpg` (on all platforms) If all of your input entries use this notation, you do not need to specify the `-b` option. This option also allows you to process entries from the output files of the `ldapsearch` tool when it uses the `-t` option.
`-A`		Non-ASCII mode: display non-ASCII values, in conjunction with the `-v` option.
`-c`		Continuous mode: errors are reported but the `ldapmodify` tool will continue processing input and performing operations. When this option is omitted, the default is to quit after reporting an error.
`-e`	*errorFile*	Invalid update statements in the input will be copied to the *errorFile* for debugging. Use with the `-c` option to correct errors when processing large LDIF input.

SSL (Secure Socket Layer) Options

The options in Table 4-3 allow you to use LDAPS (LDAP over SSL) to establish a secure connection for the update operation. These options are valid only when LDAPS has been enabled and configured in your SSL-enabled directory server. For information on certificate-based authentication and creating a certificate database for use with LDAP clients, see Chapter 11, "Implementing Security," in the *Sun ONE Directory Server Administration Guide*. See "Using Authentication" on page 86 for examples using the SSL options.

NOTE Only the `-P` option is required for server authentication. For the more secure client authentication, the `-P`, `-N`, `-K` and `-W` options are required.

Table 4-3 SSL Options for `ldapmodify`

Option	Parameter	Purpose
`-P`	*path*	Specify the path and filename of the client's certificate database. This file may be the same as the certificate database for an SSL-enabled version of Netscape™ Communicator, if available; for example: `-P /home/`*uid*`/.netscape/cert7.db`. When using the command on the same host as the directory server, you may use the server's own certificate database, for example: `-P` *installDir*`/slapd-`*serverID*`/alias/cert7.db`. Use the `-P` option alone to specify server authentication only.
`-Z`		Specify that SSL be used to provide certificate-based client authentication. This option requires the `-N` and `-W` options and any other of the SSL options needed to identify the certificate and the key database.
`-N`	*certificate*	Specify the certificate name to use for certificate-based client authentication, for example: `-N "Directory-Cert"`.
`-m`	*path*	Specify the path to the security module database. For example, `/usr/iplanet/servers/slapd-`*serverID*`/secmodule.db`. You need to specify this option only if the security module database is in a different directory from the certificate database itself.
`-K`	*keyFile*	Specify the file and path name of the client's private key database. This option may be omitted if the key database is in the location already given by the `-P` option.
`-W`	*password*	Specify the password for the client's key database given in the `-K` or `-P` options. This option is required for certificate-based client authentication.

Return Values

The `ldapmodify` tool is based on the Sun ONE LDAP SDK for C and its return values are those of the functions it uses, such as `ldap_simple_bind_s()`, `ldap_add_ext_s()`, `ldap_modify_ext_s()`, and `ldap_delete_ext_s()`. These functions return both client-side and server-side errors and codes. Table 4-4 shows the possible return values when the directory is hosted on a Sun ONE Directory Server. Other LDAP servers may send these values under different circumstances or may send different values. They may also send other result codes entirely; for example, custom result codes from a custom plug-in. For further information about result codes, see the *Sun ONE LDAP SDK for C Programming Guide.*

Table 4-4 Return Values of `ldapmodify`

Return Value	Result Code and Explanation
0 (0x00)	`LDAP_SUCCESS`: the operation was successful.
1 (0x01)	`LDAP_OPERATIONS_ERROR`: sent by Directory Server for general errors encountered by the server when processing the request.
2 (0x02)	`LDAP_PROTOCOL_ERROR`: the modify request did not comply with the LDAP protocol. Directory Server may set this error code in the results for a variety of reasons, such as encountering an error when decoding the BER-encoded request.
10 (0x0a)	`LDAP_REFERRAL`: sent by Directory Server if the specified DN is an entry not handled by the current server and if the referral URL identifies a different server to handle the entry.
16 (0x10)	`LDAP_NO_SUCH_ATTRIBUTE`: sent by Directory Server if the attribute that you want to modify (add, replace, or delete) does not exist.
19 (0x13)	`LDAP_CONSTRAINT_VIOLATION`: sent by Directory Server when improperly modifying the `userpassword` attribute, for example if the new value is shorter than the allowed minimum length.
20 (0x14)	`LDAP_TYPE_OR_VALUE_EXISTS`: sent by Directory Server when attempting to add an attribute to an entry in which the attribute already exists with the given value.
21 (0x15)	`LDAP_INVALID_SYNTAX`: sent by Directory Server if your client is modifying the schema entry and no object class or attribute type is specified.
32 (0x20)	`LDAP_NO_SUCH_OBJECT`: sent by Directory Server if the entry that you want to modify or delete does not exist.
50 (0x32)	`LDAP_INSUFFICIENT_ACCESS`: sent by Directory Server if the DN used for authentication does not have permission to write to the entry.

Table 4-4 Return Values of `ldapmodify` *(Continued)*

Return Value	Result Code and Explanation
53 (0x35)	`LDAP_UNWILLING_TO_PERFORM`: sent by Directory Server when: • The directory is read-only. • Attempting to add attributes to the special directory configuration entry. • Attempting to modify attributes in the special schema entry.
65 (0x41)	`LDAP_OBJECT_CLASS_VIOLATION`: sent by Directory Server if the modified entry does not comply with the directory schema (for example, if one or more required attributes are not specified).
67 (0x43)	`LDAP_NOT_ALLOWED_ON_RDN`: sent by Directory Server if the modified entry no longer contains attributes for each DN component.
68 (0x44)	`LDAP_ALREADY_EXISTS`: sent by Directory Server if the DN of the entry that you want to add is already present in the directory.
81 (0x51)	`LDAP_SERVER_DOWN`: the LDAP server did not receive the request or the connection to the server was lost.
82 (0x52)	`LDAP_LOCAL_ERROR`: an error occurred when receiving the results from the server.
83 (0x53)	`LDAP_ENCODING_ERROR`: BER-encoding the request is not possible.
84 (0x54)	`LDAP_DECODING_ERROR`: an error occurred when decoding the BER-encoded results from the server.
89 (0x59)	`LDAP_PARAM_ERROR`: one of the options or parameters is invalid.
90 (0x5a)	`LDAP_NO_MEMORY`: memory cannot be allocated as needed.
91 (0x5b)	`LDAP_CONNECT_ERROR`: the specified hostname or port is invalid.
92 (0x5c)	`LDAP_NOT_SUPPORTED`: the `-V 2` option is needed to access a server that only supports LDAP v2.

Command-Line Examples

The examples in this section demonstrate common uses of the `ldapmodify` tool to update the contents of a directory. All examples assume the following:

- The given bind DN has the permission to perform all operations on the selected entries.
- The directory server is located on a machine with the given *hostname*.
- The server uses the default port number 389 so you do not have to specify the port number on the search request.

- SSL is enabled for the server on the default SSL port number 636.

Adding an Entry

This example uses the `-a` option for bulk addition, so the `changetype: add` statement and keyword are not needed in the input. Instead, it contains standard LDIF entries to be added. Code Example 4-2 is the input file called `newEntry.ldif` which defines only one entry to add.

Code Example 4-2 newEntry.ldif Input File

```
dn: cn=Pete Minsky,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Pete Minsky
givenName: Pete
sn: Minsky
ou: People
ou: Marketing
uid: peterm
```

To perform this addition, launch the `ldapmodify` tool with the `-a` option and specify the input file with the `-f` option:

```
$ ldapmodify -h hostname -a -f newEntry.ldif \
             -D "uid=bjensen,dc=example,dc=com" -w bindPassword
```

Modifying an Entry

The update statement for a modification includes change records that specify the attributes to change and their new values. (See “Managing Entries From the Command Line” in Chapter 2 of the *Sun ONE Directory Server Administration Guide* for a description of this syntax.) Code Example 4-3 is the `modifyEntry.ldif` file which includes statements for adding a new attribute and modifying an existing one. The line with a single dash (-) is a separator for multiple modifications in the same entry.

Code Example 4-3 modifyEntry.ldif Input File

```
dn: cn=Pete Minsky,ou=People,dc=example,dc=com
changetype: modify
add: telephonenumber
telephonenumber: (408) 555-2468
-
replace: uid
uid: pminsky
```

To perform the operation, launch the `ldapmodify` tool and specify the filename on the command-line.

```
$ ldapmodify -h hostname -f modifyEntry.ldif \
              -D "uid=bjensen,dc=example,dc=com" -w bindPassword
```

Deleting an Entry

The update statement for a deletion requires only the DN and the `changetype`. This example shows how to enter this information as standard input on the command-line:

```
$ ldapmodify -h hostname \
             -D "uid=bjensen,dc=example,dc=com" \
             -w bindPassword \
             dn: cn=Pete Minsky,ou=People,dc=example,dc=com \
              changetype: delete ^D
```

Using Authentication

There are two levels of authentication that the directory server may enforce on clients such as the `ldapmodify` tool: server authentication and client authentication. In server authentication, the server accepts connections only from clients that have a trusted certificate. In the stronger client authentication the client must sign the certificate with a password-protected private key.

NOTE In both cases, use the `-p` option to specify the directory server's SSL port. All other non-SSL options retain their original meaning and may be used as necessary.

Using Server Authentication

To run the `ldapmodify` tool with server authentication, use only the `-P` SSL option [as discussed in “SSL (Secure Socket Layer) Options” on page 82] on the command-line, in addition to other common options.

```
$ ldapmodify -h hostname -p 636 -f LDIFfile \
             -D "uid=bjensen,dc=example,dc=com" -w bindPassword \
             -P /home/bjensen/certs/cert.db
```

Using Client Authentication

To perform an update with client authentication, you must give all SSL options [as discussed in “SSL (Secure Socket Layer) Options” on page 82] on the command-line, in addition to other common options.

```
$ ldapmodify -h hostname -p 636 -f LDIFfile \
             -Z -P /home/bjensen/security/cert.db -N "bjscert" \
              -K /home/bjensen/security/key.db -W KeyPassword
```

CAUTION Do not use the `-D` and `-w` common options with client authentication, as the bind operation will use the authentication credentials specified with `-D` and `-w` instead of the certificate credentials desired.

Chapter 5

The ldapdelete Tool

The `ldapdelete` tool is a simple command for deleting entries in a Lightweight Directory Access Protocol (LDAP) directory. This chapter provides instructions on how to use the `ldapdelete` tool. It contains the following sections:

- Overview
- Command Usage
- Return Values
- Common Use Examples

Overview

In its simplest form, `ldapdelete` takes distinguished names (DNs), defined using the LDAP Data Interchange Format (LDIF), as input and deletes the corresponding entries. The DNs can be input on the command-line, via standard input or they can be culled from a file for bulk processing. (See “Common Use Examples” on page 96 for information on how to use all three options.)

`ldapdelete` is also provided with Sun™ ONE Directory Server in the *DirectoryServer_base*`/shared/bin` directory. However, the DSRK and its updates should include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory.

CAUTION If you use the Solaris™ operating environment, there may be an older version of `ldapdelete` in `/usr/bin`. Be sure your path is set to use the latest version in *DSRK_base*`/bin/dsrk52`.

Command Usage

The `ldapdelete` command binds to the given directory server and deletes each LDAP entry defined by a DN in the input.

NOTE In order to delete an entry in a directory server, the DN used for binding and authentication must have the correct permissions.

Only leaf entries, which do not have any children, may be deleted from a directory using `ldapdelete`. For example, when deleting a subtree representing an organizational unit, you must first delete all the entries it contains before deleting the entry representing the organizational unit. Thus, when deleting DNs listed in a file, `ldapdelete` will process each delete operation separately, in the order they are defined in the file. Therefore, DNs representing leaf entries must be listed before the DNs for their parent entries.

Syntax

The syntax of `ldapdelete` on the command-line can take any of these forms:

```
ldapdelete [ options ]
ldapdelete [ options ] "DN" ...
ldapdelete [ options ] < DNfile
ldapdelete [ options ] -f DNfile
```

Where:

- *options* are the command-line options and their parameters described in “Options” on page 91.
- *DN* ... is a space-separated list of DNs to delete. Each DN should be enclosed in double quote marks (" ") for the shell interpreter. The list of DNs is not required if you give the DNs in a *DNfile*.
- *DNfile* is a text file containing one DN per line. Do not use quote marks in this file because each line is taken literally.

In its first and second forms, the tool will expect you to type one or more DNs to the standard input. Once you enter all DNs and the EOF (end-of-file) marker, `ldapdelete` will process your input and perform all operations.

NOTE The EOF marker is platform dependent:

- Type Control-D (^d) on most UNIX systems.
- Type Control-Z (^z) and press Enter on Windows® NT®.

The last two forms take a *DNfile* as input. The first of the two uses < (the *less than* symbol) to take the input from the specified file instead of the keyboard. The final syntax does the same by using the `-f` option. Some samples of syntax and update statements are given in "Common Use Examples" on page 96.

NOTE For general information on LDIF, see Appendix E, "LDAP Data Interchange Format," in the *Sun ONE Directory Server Reference Manual*). Additional information on LDIF and update statements is in "Managing Entries From the Command Line" in Chapter 2 of the *Sun ONE Directory Server Administration Guide*.

Options

The `ldapdelete` tool has three types of options:

- Common Options
- Input And Output Options
- SSL (Secure Socket Layer) Options

The following sections detail these options. The `ldapdelete -H` command and option when run on the command-line will display brief descriptions of all the command-line options.

Common Options

The common options listed in Table 5-1 control the binding and general behavior of the `ldapdelete` command.

Table 5-1 Common Options for `ldapdelete`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. When this option is omitted, the default is `localhost`.
`-p`	*port*	Specify the port number for accessing the directory server host. The default is 389 normally and 636 when the SSL options are used.

Table 5-1 Common Options for `ldapdelete` *(Continued)*

Option	Parameter	Purpose
`-D`	*bindDN*	Specify a bind DN for accessing your directory, usually in double quotes ("") for the shell. If the bind DN and its password are omitted, the tool will use anonymous binding. The bind DN determines what entries may be deleted, according to the DN's access permissions.
`-w`	*password*	Specify the password for the bind DN. **CAUTION: Specifying the password on the command-line is a possible security risk.**
`-w`	`-`	Type the password for the bind DN when prompted. This is the most secure way of specifying the password.
`-j`	*filename*	Specify a file containing the password for the bind DN. Use this option in scripts and place the password in a secure file to protect the password. This option is mutually exclusive with the `-w` option.
`-f`	*DNfile*	Give the name of a file containing the DNs of entries to be deleted. The DNs should be listed one per line in this file, in the order in which they must be deleted. When this option is omitted, `ldapdelete` will read DNs directly from the standard input.
`-V`	*version*	Specify the LDAP protocol version number to be used for the delete operation, either 2 or 3. LDAP v3 is the default; only specify LDAP v2 when connecting to servers that do not support v3.
`-Y`	*proxyDN*	Specify the proxy DN to use for the delete operation, usually in double quotes ("") for the shell. For more information about proxy authorization, see Chapter 6, "Managing Access Control," in the *Sun ONE Directory Server Administration Guide*.
`-M`		Manage smart referrals: when they are the target of the operation, delete the actual entry containing the referral instead of the entry obtained by following the referral. For more information, see "Creating Smart Referrals" in Chapter 2 of the *Sun ONE Directory Server Administration Guide*.
`-O`	*hopLimit*	(Capital letter O) Specify the maximum number of referral hops to follow while finding an entry to delete. By default, there is no limit.
`-R`		Specify that referrals should *not* be followed. By default, referrals are followed automatically.
`-v`		Verbose output mode: the tool will display additional information about the operations it performs.
`-n`		No-op mode: use with the `-v` option to show what the tool would do with the given input but do not perform the delete operations.

Table 5-1 Common Options for `ldapdelete` *(Continued)*

Option	Parameter	Purpose
`-0` (zero)		Allow runtime library version mismatches. When this option is omitted, the default behavior is to assert that the revision number of the LDAP API is greater than or equal to that used to compile the tool. Also, if the API library and the tool have the same vendor name, the tool will also assert that the vendor version number of the API is greater than or equal to that used to compile the tool. This information is based on the contents of the `LDAPAPIInfo` structure. (See the *Sun ONE LDAP SDK for C Programming Guide*.)
`-H`		Display the usage help text that briefly describes all options.

Input And Output Options

The input and output options listed in Table 5-2 control how `ldapdelete` processes input files and handles errors.

Table 5-2 Input and Output Options for `ldapdelete`

Option	Parameter	Purpose
`-i`	*locale*	Specify the character set to use for command-line input. The default is the character set specified in the `LANG` environment variable. This option can be used to convert from the specified character set to UTF8, thus overriding the `LANG` setting. Using this argument, you can also input the bind DN and target DNs in a specific character set. The `ldapdelete` tool converts the input before it processes the search request. For example, `-i no` indicates that the bind DN and target DNs are provided in Norwegian. This option affects only the command-line input; that is, if you specify a file containing DNs (with the `-f` option), `ldapdelete` will not convert the data in the file.
`-k`	*path*	Specify the path to a directory containing conversion routines. These routines are used if you wish to specify a locale that is not supported by default by your directory server. For more information, see Appendix C, “Directory Internationalization” in the *Sun ONE Directory Server Reference Manual*.
`-c`		Continuous mode: when errors are reported, the `ldapdelete` tool will continue processing input and performing operations. When this option is omitted, the default is to quit after reporting an error.

SSL (Secure Socket Layer) Options

The options in Table 5-3 allow you to use LDAPS (LDAP over SSL) to establish a secure connection for the delete operation. These options are valid only when LDAPS has been enabled and configured in your SSL-enabled directory server. For information on certificate-based authentication and creating a certificate database for use with LDAP clients, see Chapter 11, "Implementing Security," in the *Sun ONE Directory Server Administration Guide*. See "Using Authentication" on page 97 for examples using the SSL options.

NOTE Only the `-P` option is required for server authentication. For a more secure client authentication, the `-P`, `-N`, `-K` and `-W` options are required.

Table 5-3 SSL Options for `ldapdelete`

Option	Parameter	Purpose
`-P`	*path*	Specify the path and filename of the client's certificate database. This file may be the same as the certificate database for an SSL-enabled version of Netscape™ Communicator, if available; for example: `-P /home/`*uid*`/.netscape/cert7.db`.
		When using the command on the same host as the directory server, you may use the server's own certificate database, for example: `-P` *installDir*`/slapd-`*serverID*`/alias/cert7.db`.
		Use the `-P` option alone to specify server authentication only.
`-Z`		Specify that SSL be used to provide certificate-based client authentication. This option requires the `-N` and `-W` options and any other of the SSL options needed to identify the certificate and the key database.
`-N`	*certificate*	Specify the certificate name to use for certificate-based client authentication, for example: `-N "Directory-Cert"`.
`-m`	*path*	Specify the path to the security module database. For example, `/usr/iplanet/servers/slapd-`*serverID*`/secmodule.db`. You need to specify this option only if the security module database is in a different directory from the certificate database itself.
`-K`	*keyFile*	Specify the file and path name of the client's private key database. This option may be omitted if the key database is in the location already given by the `-P` option.
`-W`	*password*	Specify the password for the client's key database given in the `-K` or `-P` options. This option is required for certificate-based client authentication.

Return Values

The `ldapdelete` tool is based on the Sun ONE LDAP SDK for C and its return values are those of the functions it uses, such as `ldap_simple_bind_s()` and `ldap_delete_ext_s()`. These functions return both client-side and server-side errors and codes. Table 5-4 shows the possible return values when the directory is hosted on a Sun ONE Directory Server. Other LDAP servers may send these values under different circumstances or may send different values. They may also send other result codes entirely; for example, custom result codes from a custom plug-in. For further information about result codes, see the *Sun ONE LDAP SDK for C Programming Guide*.

Table 5-4 Return Values of `ldapdelete`

Return Value	Result Code and Explanation
0 (0x00)	`LDAP_SUCCESS`: the operation was successful.
1 (0x01)	`LDAP_OPERATIONS_ERROR`: sent by Directory Server for general errors encountered by the server when processing the request.
2 (0x02)	`LDAP_PROTOCOL_ERROR`: the delete request did not comply with the LDAP protocol. Directory Server may send this error code in the results for a variety of reasons, such as encountering an error when decoding the BER-encoded request.
10 (0x0a)	`LDAP_REFERRAL`: sent by Directory Server if the specified DN is an entry not handled by the current server and the referral URL identifies a different server to handle the entry.
32 (0x20)	`LDAP_NO_SUCH_OBJECT`: sent by Directory Server if the entry that you want deleted does not exist and no referral URLs are available.
50 (0x32)	`LDAP_INSUFFICIENT_ACCESS`: sent by Directory Server if the DN used for authentication does not have permission to write to the entry.
53 (0x35)	`LDAP_UNWILLING_TO_PERFORM`: sent by Directory Server if the database is read-only.
66 (0x42)	`LDAP_NOT_ALLOWED_ON_NONLEAF`: sent by Directory Server if the entry that you want deleted has child entries (i.e.: if this entry is a parent entry to other entries).
81 (0x51)	`LDAP_SERVER_DOWN`: the LDAP server did not receive the request or the connection to the server was lost.
82 (0x52)	`LDAP_LOCAL_ERROR`: an error occurred when receiving the results from the server.
83 (0x53)	`LDAP_ENCODING_ERROR`: BER-encoding the request is not possible.
84 (0x54)	`LDAP_DECODING_ERROR`: an error occurred when decoding the BER-encoded results from the server.
89 (0x59)	`LDAP_PARAM_ERROR`: one of the options or parameters is invalid.

Table 5-4 Return Values of `ldapdelete` *(Continued)*

Return Value	Result Code and Explanation
90 (0x5a)	`LDAP_NO_MEMORY`: memory cannot be allocated as needed.
91 (0x5b)	`LDAP_CONNECT_ERROR`: the specified hostname or port is invalid.
92 (0x5c)	`LDAP_NOT_SUPPORTED`: the `-V 2` option is needed to access a server that only supports LDAP v2.

Common Use Examples

The examples in this section demonstrate common uses of the `ldapdelete` tool. All examples assume the following context:

- The given bind DN has the permission to perform delete operations on the selected entries.
- The directory server is located on a machine with the given *hostname*.
- The server uses port number 389. Because this is the default port, you do not have to specify the port number on the search request.
- SSL is enabled for the server on port 636 (the default SSL port number).

Using the Command Line

The simplest usage of the tool is to specify the target DNs on the command-line, making sure to enclose them in double quote marks for the shell. In addition, special characters such as commas must be escaped with a backslash (\) when they appear in components of a DN on the command-line. In the following example, the user `bjensen` wishes to delete an entry in the "Company Bolivia, S.A." subtree of the directory:

```
$ ldapdelete -h hostname -D "uid=bjensen,dc=Company,dc=com" -w bindPassword
"cn=Lucia Fuentes,ou=People,o=Company Bolivia\,S.A."
```

Using the Standard Input

Using the `-v` and `-c` options, you can enter DNs interactively through the standard input. In Code Example 5-1 on page 97, user `bjensen` wishes to remove an entry but at first enters the DN incorrectly. `bjensen` is typing the lines in bold.

Code Example 5-1 Using lapdelete via Standard Input

```
$ ldapdelete -h hostname -v -c -D "uid=bjensen,dc=example,dc=com"
-w bindPassword

ldapdelete: started Thu Jun 14 11:34:17 2001
ldap_init( host, 389 )

$ cn=Pete Mimsky,ou=People,dc=example,dc=com

deleting entry cn=Pete Mimsky,ou=People,dc=example,dc=com
ldap_delete: No such object

$ cn=Pete Minsky,ou=People,dc=example,dc=com

deleting entry cn=Pete Minsky,ou=People,dc=example,dc=com
entry removed

$ ^D
```

Using a DN File

For bulk operations, list all of the DNs to delete in a text file. This text file should specify each DN on a separate line. The `ldapdelete` command will perform a delete operation on each entry, in the order in which they appear in the file. For example:

```
cn=Pete Minsky,ou=People,dc=example,dc=com
cn=Sue Jacobs,ou=People,dc=example,dc=com
```

Use the `-f` option to specify this *DNfile* on the command-line. Use the `-c` option so that bulk processing will continue even if errors are encountered.

```
$ ldapdelete -h hostname -c -f DNfile -D "uid=bjensen,dc=example,dc=com" -w
bindPassword
```

Using Authentication

There are two levels of authentication that the directory server may enforce on clients such as the `ldapdelete` tool: server authentication and client authentication. In server authentication, the server only accepts connections from clients that have a trusted certificate. In the stronger client authentication, the client must sign the certificate with a password-protected private key.

NOTE In both cases, use the `-p` option to specify the directory server's SSL port. All other non-SSL options retain their original meaning and may be used as necessary.

Using Server Authentication

To run the `ldapdelete` tool with server authentication, give only the `-P` SSL option [as discussed in "SSL (Secure Socket Layer) Options" on page 94] on the command-line, in addition to other options.

```
$ ldapdelete -h hostname -p 636 -f DNfile \
             -D "uid=bjensen,dc=siroe,dc=com" -w bindPassword \
              -P /home/bjensen/certs/cert.db
```

Using Client Authentication

To perform an update with client authentication, you must give all SSL options [as discussed in "SSL (Secure Socket Layer) Options" on page 94], on the command-line in addition to other common options.

```
$ ldapdelete -h hostname -p 636 -f DNfile -Z \
             -P /home/bjensen/security/cert.db -N "bjscert" \
             -K /home/bjensen/security/key.db -W KeyPassword
```

CAUTION Do not use the `-D` and `-w` common options with client authentication, as the bind operation will use the authentication credentials specified with `-D` and `-w` instead of the certificate credentials desired.

Chapter 6

The ldapcompare Tool

The `ldapcompare` tool compares an attribute value against the contents of a given directory entry. It compares either a textual value or a binary value, providing a simple interface to check data against directory contents. This chapter provides instructions on how to use the `ldapcompare` tool. It contains the following sections:

- Overview
- Command Usage
- Return Values
- Command-Line Examples

Overview

The `ldapcompare` tool compares one attribute value with the same attribute's value in one or more entries of a directory. Standard, textual values can be entered directly on the command-line; binary attribute values may be entered on the command-line with base-64 encoding or stored in a file referenced using the URL syntax.

TIP To compare the contents of two directories, use `ldapcmp` described in Chapter 7.

`ldapcompare` is also provided with Sun™ ONE Directory Server in the *DirectoryServer_base*`/shared/bin` directory. However, the DSRK and its updates should include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory.

CAUTION If you use the Solaris™ operating environment, there may be an older version of `ldapcompare` in `/usr/bin`. Be sure your path is set to use the latest version in *DSRK_base*`/bin/dsrk52`.

Command Usage

`ldapcompare` supports the common options of the other LDAP commands, such as managing referrals, handling locales, and providing SSL-based security. By default, it will print out the success of the comparisons, stopping after the first failed comparison. The command-line options can be used to control this behavior and other aspects of the input and output.

Syntax

The syntax of the `ldapcompare` tool on the command-line can take any of these forms:

`ldapcompare [` *options* `] '`*attribute*`:`*value*`' [ "`*targetDN*`" ... | -f` *DNfile* `]`

`ldapcompare [` *options* `] '`*attribute*`::`*base64value*`' [ "`*targetDN*`" ... | -f` *DNfile* `]`

`ldapcompare [` *options* `] '`*attribute*`:<`*fileURL*`' [ "`*targetDN*`" ... | -f` *DNfile* `]`

Where:

- *options* are the command-line options described in “Options” on page 100.
- *attribute* is the type name of the attribute, followed by one of the three ways to specify its comparative value. The attribute type name and value string should be enclosed in single quotes (**'**) for the shell.
- *targetDN* is the distinguished name (DN) or list of DNs in which to search for the given attribute and compare its value.
- *DNfile* is a file with a list of DNs, one per line, to search for the given attribute and compare its value.

Options

The `ldapcompare` tool has three types of options:

- Common Options

- Input And Output Options
- SSL (Secure Socket Layer) Options

The following sections detail these options. The `ldapcompare -H` command and option when run on the command-line will display brief descriptions of the command syntax, options, and parameters.

Common Options

The common options listed in Table 6-1 control the binding and general behavior of the `ldapcompare` command.

Table 6-1 Common Options for `ldapcompare`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. When this option is omitted, the default is `localhost`.
`-p`	*port*	Specify the port number for accessing the directory server host. The default is 389 normally and 636 when the SSL options are used.
`-D`	*bindDN*	Specify a bind DN for accessing your directory with simple authentication, usually in double quotes ("") for the shell. If the bind DN and its password are omitted, the tool will use anonymous binding. The bind DN determines what entries may be accessed, according to the DN's access permissions.
`-w`	*password*	Specify the password for the bind DN. **CAUTION: Specifying the password on the command-line is a possible security risk.**
`-w`	`-`	Type the password for the bind DN when prompted. This is the most secure way of specifying the password.
`-j`	*filename*	Specify a file containing the password for the bind DN. Use this option in scripts and place the password in a secure file to protect the password. This option is mutually exclusive with the `-w` option.
`-f`	*DNfile*	Give the name of a file containing the DNs of entries to be compared. The DNs should be listed one per line in this file, each line being taken as the entire literal DN. Do not use quotes.
`-V`	*version*	Specify the LDAP protocol version number to be used for the comparison, either 2 or 3. LDAP v3 is the default; only specify LDAP v2 when connecting to servers that do not support v3.
`-Y`	*proxyDN*	Specify the proxy DN to use for the comparison, usually in double quotes ("") for the shell. For more information about proxy authorization, see Chapter 6, "Managing Access Control," in the *Sun ONE Directory Server Administration Guide*.

Table 6-1 Common Options for `ldapcompare` *(Continued)*

Option	Parameter	Purpose
`-M`		Manage smart referrals: when they are the target of the comparison, compare values in the actual entry containing the referral instead of the entry obtained by following the referral. For more information, see "Creating Smart Referrals" in Chapter 2 of the *Sun ONE Directory Server Administration Guide*.
`-O`	*hopLimit*	(Capital letter O) Specify the maximum number of referral hops to follow while finding an entry to compare. By default, there is no limit.
`-R`		Specify that referrals should *not* be followed. By default, referrals are followed automatically.
`-v`		Verbose output mode: the tool will display additional information about the operations it performs.
`-n`		No-op mode: use with the `-v` option to show what the tool would do with the given input but do not perform the comparison.
`-0` (zero)		Allow runtime library version mismatches. When this option is omitted, the default behavior is to assert that the revision number of the LDAP API is greater than or equal to that used to compile the tool. Also, if the API library and the tool have the same vendor name, the tool will also assert that the vendor version number of the API is greater than or equal to that used to compile the tool. This information is based on the contents of the `LDAPAPIInfo` structure. (See the *Sun ONE LDAP SDK for C Programming Guide*.)
`-H`		Display the usage help text that briefly describes all options.

Input And Output Options

The input and output options listed in Table 6-2 control how `ldapcompare` processes input files and handles errors.

Table 6-2 Input and Output Options for `ldapcompare`

Option	Parameter	Purpose
`-i`	*locale*	Specify the character set to use for command-line input. The default is the character set specified in the `LANG` environment variable. This option can be used to convert from the specified character set to UTF8, thus overriding the `LANG` setting. You can also input a specific character set for the bind DN, base DN, and the search filter pattern. The `ldapcompare` tool converts the input from these arguments before it processes the request. For example, `-i no` indicates that the bind DN and target DNs are provided in Norwegian. This argument only affects the command-line input. A DN file (specified with the `-f` option) will not be converted by `ldapcompare`.

Table 6-2 Input and Output Options for `ldapcompare` *(Continued)*

Option	Parameter	Purpose
`-k`	*path*	Specify the path to a directory containing conversion routines. These routines are used if you wish to specify a locale that is not supported by default by your directory server. For more information, see Appendix C, "Directory Internationalization" in the *Sun ONE Directory Server Reference Manual*.
`-c`		Continuous mode: errors are reported but the `ldapcompare` tool will continue processing input and performing operations. When this option is omitted, the default behavior is to quit after reporting an error.
`-q`		Quiet mode: information and results of comparisons are not displayed in the output, however LDAP errors still are.

SSL (Secure Socket Layer) Options

The options in Table 6-3 allow you to use LDAPS (LDAP over SSL) to establish a secure connection for the compare operation. These options are valid only when LDAPS has been turned on and configured in your SSL-enabled directory server. For information on certificate-based authentication and creating a certificate database for use with LDAP clients, see Chapter 11, "Implementing Security," in the *Sun ONE Directory Server Administration Guide*.

NOTE Only the `-P` option is required for server authentication. For the more secure client authentication, the `-P`, `-N`, `-K` and `-W` options are required.

Table 6-3 SSL Options for `ldapcompare`

Option	Parameter	Purpose
`-P`	*path*	Specify the path and filename of the client's certificate database. This file may be the same as the certificate database for an SSL-enabled version of Netscape™ Communicator, if available; for example: `-P /home/`*uid*`/.netscape/cert7.db`.
		When using the command on the same host as the directory server, you may use the server's own certificate database, for example: `-P` *installDir*`/slapd-`*serverID*`/alias/cert7.db`.
		Use the `-P` option alone to specify server authentication only.
`-Z`		Specify that SSL be used to provide certificate-based client authentication. This option requires the `-N` and `-W` options and any other of the SSL options needed to identify the certificate and the key database.

Table 6-3 SSL Options for `ldapcompare` *(Continued)*

Option	Parameter	Purpose
`-N`	*certificate*	Specify the certificate name to use for certificate-based client authentication, for example: `-N "Directory-Cert"`.
`-m`	*path*	Specify the path to the security module database. For example, `/usr/iplanet/servers/slapd-`*serverID*`/secmodule.db`. You need to specify this option only if the security module database is in a different directory from the certificate database itself.
`-K`	*keyFile*	Specify the file and path name of the client's private key database. This option may be omitted if the key database is in the location already given by the `-P` option.
`-W`	*password*	Specify the password for the client's key database given in the `-K` or `-P` options. This option is required for certificate-based client authentication.

Return Values

The `ldapcompare` tool is based on the Sun ONE LDAP SDK for C and its return values are those of the functions it uses, such as `ldap_simple_bind_s()`, `ldap_compare_ext()`, and `ldap_result()`. These functions return both client-side and server-side errors and codes. Table 6-4 shows the possible return values when the directory is hosted on a Sun ONE Directory Server. Other LDAP servers may send these values under different circumstances or may send different values. They may also send other result codes entirely; for example, custom result codes from a custom plug-in. Under most conditions though, the `ldapcompare` tool returns one of the following integer values:

- 6 (`LDAP_COMPARE_TRUE`) when the comparison is successful.
- 5 (`LDAP_COMPARE_FALSE`) when the comparison is unsuccessful.
- Some other value when an error occurs.

For further information about result codes, see the *Sun ONE LDAP SDK for C Programming Guide*.

Table 6-4 Return Values of `ldapcompare`

Return Value	Result Code and Explanation
1 (0x01)	`LDAP_OPERATIONS_ERROR`: sent by Directory Server for general errors encountered by the server when processing the request.

Table 6-4 Return Values of `ldapcompare` *(Continued)*

Return Value	Result Code and Explanation
2 (0x02)	`LDAP_PROTOCOL_ERROR`: Directory Server may send this error code in the results for a variety of reasons, such as encountering an error when decoding the BER-encoded request.
3 (0x03)	`LDAP_TIMELIMIT_EXCEEDED`: sent by Directory Server if a comparison search exceeded the default time limit for an operation on the server.
4 (0x04)	`LDAP_SIZELIMIT_EXCEEDED`: sent by Directory Server if a comparison search found more results than the default maximum allowed by the server.
5 (0x05)	`LDAP_COMPARE_FALSE`: returned by the tool when the operation is successful but the comparison fails.
6 (0x06)	`LDAP_COMPARE_TRUE`: returned by the tool when the operation is successful and the comparison holds.
10 (0x0a)	`LDAP_REFERRAL`: sent by Directory Server if the given base DN is an entry not handled by either server and if the referral URL identifies a different server to handle the entry.
11 (0x0b)	`LDAP_ADMINLIMIT_EXCEEDED`: sent by Directory Server if a comparison search found more results than the limit specified by the `lookthroughlimit` directive in the `slapd.conf` configuration file. If not specified in the configuration file, the default limit is 5000.
32 (0x20)	`LDAP_NO_SUCH_OBJECT`: the given base DN cannot be found in the directory server and no referral URL is available.
50 (0x32)	`LDAP_INSUFFICIENT_ACCESS`: sent by the directory server if the DN used for authentication does not have permission to read from the directory.
81 (0x51)	`LDAP_SERVER_DOWN`: the directory server did not respond to the comparison operations or the connection was lost.
82 (0x52)	`LDAP_LOCAL_ERROR`: an error occurred when receiving the results from the server.
83 (0x53)	`LDAP_ENCODING_ERROR`: the request could not be BER-encoded.
84 (0x54)	`LDAP_DECODING_ERROR`: an error occurred when decoding the BER-encoded results from the server.
89 (0x59)	`LDAP_PARAM_ERROR`: one of the options or parameters is invalid.
90 (0x5a)	`LDAP_NO_MEMORY`: memory cannot be allocated as needed.
91 (0x5b)	`LDAP_CONNECT_ERROR`: a specified hostname or port is invalid.
92 (0x5c)	`LDAP_NOT_SUPPORTED`: the `-V 2` option is needed to access a server that only supports LDAP v2.

Command-Line Examples

The following examples demonstrate the three types of values on which the tool will perform comparisons. All examples assume the following context:

- The server uses the default port number 389.
- The bind DN and password for simple authentication are omitted, so `ldapcompare` will use anonymous binding to perform operations.

Comparing a Text Value

In this example, `ldapcompare` takes a textual value from the command-line and compares it to an attribute in the given DN.

```
% ldapcompare -h phonebook.example.com 'givenname:Barbara'
"uid=bjensen,ou=People,dc=example,dc=com"

comparing type: "givenname" value: "Barbara" in entry
"uid=bjensen,ou=People,dc=example,dc=com" compare TRUE
```

Comparing a Binary Value

The following two examples use base-64 binary encoding. First, the comparative value is given directly in base-64 binary encoding, allowing you to compare binary values from some other output, for example, a script. This example also shows the output for multiple entry comparisons (including one which fails for authentication reasons) and uses the `-c` option to ensure all entries are compared.

```
% ldapcompare -h phonebook.example.com -c 'cn:: d29vZiAK'
"uid=tmorgan,ou=People,dc=example,dc=com" "dc=example,dc=com"

comparing type: "cn" value: "woof" in entry
"uid=tmorgan,ou=People,dc=example,dc=com" compare FALSE

comparing type: "cn" value: "woof" in entry "dc=example,dc=com"
ldap_compare: Insufficient access
```

Finally, the binary value to compare is a file referenced by a URL.

```
% ldapcompare -h phonebook.example.com
'usercertificate;binary:<file:/tmp/mycert'
"uid=bjensen,ou=People,dc=example,dc=com"

comparing type: "usercertificate;binary" value: "NOT ASCII (777 bytes)"
in entry "uid=bjensen,ou=People,dc=siroe,dc=com" compare TRUE
```

Chapter 7

The ldapcmp Tool

The `ldapcmp` tool compares the contents of a single Lightweight Directory Access Protocol (LDAP) entry (or an entire LDAP subtree) that is present in two directories. It detects entries that do not appear in both directories and the attribute differences in those that do. This chapter provides instructions on how to use the `ldapcmp` tool. It contains the following sections:

- Overview
- Command Usage
- Return Values
- Command-Line Examples

Overview

The `ldapcmp` tool compares parallel entries or subtrees stored in two different directories. It detects entries that do not appear in both directories and attribute differences in entries that do appear in both directories.

`ldapcmp` is also provided with Sun™ ONE Directory Server in the *DirectoryServer_base*`/shared/bin` directory. However, the DSRK and its updates should include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory.

NOTE To compare one attribute value with the same attribute's value in one or more entries of a directory, use the `ldapcompare` tool described in Chapter 6, "The ldapcompare Tool."

Command Usage

Each ldapcmp search returns all attributes for all entries in the given scope of the given base distinguished name (DN). The ldapcmp tool then compares these search results and reports the differences as in Table 7-1.

Table 7-1 How ldapcmp Reports Comparision Results

Comparison Result	How It Appears
Entries that only appear in the first directory	`1only:` *DN*
Entries that only appear in the second directory	`2only:` *DN*
Entries whose DN appears in both directories but whose attributes or attribute values are different	*matchingDN* different: *missingAttributeName* 1 *or* 2: *attributeValueWherePresent* different: *differingAttributeName* 1: *valueInDirectory1* 2: *valueInDirectory2*

ldapcmp supports the common options of the Lightweight Directory Access Protocol (LDAP) commands, such as managing referrals, handling locales, and providing SSL-based security.

Syntax

The syntax of the ldapcmp tool on the command-line takes the following form:

```
ldapcmp -h host1 -p port1 [ -h host2 -p port2 ] -b "baseDN" [ options ]
```

Where:

- *host1, port1, host2, port2* are the hostnames and port numbers for the two directories you wish to compare. The first host and port correspond to directory 1 in the output, the second to directory 2. If the second host and port are omitted, port 389 on the localhost will be used by default.
- *baseDN* is the base for the comparison, usually enclosed in double quotes ("") for the shell. The -b *baseDN* parameter may be omitted if the LDAP_BASEDN environment variable is set.
- *options* are the command-line options and their parameters described in “Options” on page 109.

NOTE Except for the host and port options, all options must apply to both directories being compared. For example, both directory servers must accept the same certificate when using the security options.

Options

The `ldapcmp` tool has three types of options:

- CommonOptions
- Input and Output Options
- SSL (Secure Socket Layer) Options

The following sections detail these options. The `ldapmodify -H` command and option when run on the command-line will display text that briefly describes all of the command-line options.

CommonOptions

The common options listed in Table 7-2 control the binding and general behavior of the `ldapcmp` command.

Table 7-2 Common Options for `ldapcmp`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of a directory server. This option may be given twice to specify the two target directories for the comparison. If it is only given once, the default for the second host is `localhost`. If it is not specified at all, the default is `localhost` for both.
`-p`	*port*	Specify the port number for accessing a directory server host. This option may be given twice to specify the port for each directory server. When either occurrence is omitted, the default is 389 normally and 636 when the SSL options are used. Note that the first occurrence of this option specifies the port for the first host, even if it appears *after* the second hostname on the command-line.
`-D`	*bindDN*	Specify a bind DN for accessing both directories, usually in double quotes (`""`) for the shell. If the bind DN and its password are omitted, the tool will use anonymous binding. The bind DN determines what entries and attributes will appear in the comparison results, according to the DN's search permissions.
`-w`	*password*	Specify the password for the bind DN. **CAUTION: Specifying the password on the command-line is a possible security risk.**

Table 7-2 Common Options for `ldapcmp` *(Continued)*

Option	Parameter	Purpose
`-w`	`-`	Type the password for the bind DN when prompted in the terminal window. This is the most secure way of specifying the password.
`-j`	*filename*	Specify a file containing the password for the bind DN. Use this option in scripts and place the password in a secure file to protect the password. This option is mutually exclusive with the `-w` option.
`-b`	*baseDN*	Specify the base DN for the comparison, usually in double quotes (`""`) for the shell. You may omit this option if you specify the base DN in the `LDAP_BASEDN` environment variable.
`-s`	*scope*	Specify the scope of the comparison. Use this option to restrict the number of entries being compared. The *scope* parameter may have one of the following values: • `base` - For comparing only the base entry. • `one` - For comparing only the children of the base entry. • `sub` - For comparing the base entry and all its descendants. This is the default if the `-s` option is omitted.
`-V`	*version*	Specify the LDAP protocol version number to be used for search operations, either 2 or 3. LDAP v3 is the default; only specify LDAP v2 when connecting to servers that do not support v3.
`-Y`	*proxyDN*	Specify the proxy DN to use for search operations, usually in double quotes (`""`) for the shell. For more information about proxy authorization, see Chapter 6, “Managing Access Control,” in the *Sun ONE Directory Server Administration Guide*.
`-M`		Manage smart referrals: when they are part of the comparison searches, return the actual entry containing the referral instead of the entry obtained by following the referral. For more information, see “Creating Smart Referrals” in Chapter 2 of the *Sun ONE Directory Server Administration Guide*.
`-O`	*hopLimit*	(Capital letter O) Specify the maximum number of referral hops to follow when performing comparison searches.
`-R`		Specify that referrals should *not* be followed. By default, referrals are followed automatically during comparison searches.
`-v`		Verbose output mode: the tool will display additional information about binding to the directory servers, searching the two directories, and comparing the search results.
`-n`		No-op mode: use with the `-v` option to show what the tool would do with the specified input but do not actually perform the searches.

Table 7-2 Common Options for `ldapcmp` *(Continued)*

Option	Parameter	Purpose
`-0` (zero)		Allow runtime library version mismatches. When this option is omitted, the default behavior is to assert that the revision number of the LDAP API is greater than or equal to that used to compile the tool. Also, if the API library and the tool have the same vendor name, the tool will also assert that the vendor version number of the API is greater than or equal to that used to compile the tool. This information is based on the contents of the `LDAPAPIInfo` structure. (See the *Sun ONE LDAP SDK for C Programming Guide*.)
`-H`		Display the usage help text that briefly describes all options.

Input and Output Options

The input and output options given in Table 7-3 control how the `ldapcmp` results are sorted and presented.

Table 7-3 Input and Output Options for `ldapcmp`

Option	Parameter	Purpose
`-i`	*locale*	Specify the character set to use for the `-f` *LDIFfile* or standard input. The default is the character set specified in the `LANG` environment variable. You might want to use this option to perform the conversion from the specified character set to UTF8, thus overriding the `LANG` setting.
`-k`	*path*	Specify the path to a directory containing conversion routines. These routines are used if you wish to specify a sorting language that is not supported by default by your directory server. For more information, see Appendix C, “Directory Internationalization” in the *Sun ONE Directory Server Reference Manual.*

SSL (Secure Socket Layer) Options

The options in Table 7-4 allow you to use LDAPS (LDAP over SSL) to establish a secure connection for the update operation. These options are valid only when LDAPS has been enabled and configured in your SSL-enabled directory server. For information on certificate-based authentication and creating a certificate database for use with LDAP clients, see Chapter 11, “Implementing Security,” in the *Sun ONE Directory Server Administration Guide.*

Table 7-4 SSL Options for `ldapcmp`

Option	Parameter	Purpose
`-P`	*path*	Specify the path and filename of the client's certificate database. This file may be the same as the certificate database for an SSL-enabled version of Netscape™ Communicator, if available; for example: `-P /home/`*uid*`/.netscape/cert7.db`. When using the command on the same host as the directory server, you may use the server's own certificate database, for example: `-P` *installDir*`/slapd-`*serverID*`/alias/cert7.db`.
		Use the `-P` option alone to specify server certification only.
`-Z`		Specify that SSL be used to provide certificate-based client authentication. This option requires the `-N` and `-W` options and any other of the SSL options needed to identify the certificate and the key database.
`-N`	*certificate*	Specify the certificate name to use for certificate-based client authentication, for example: `-N "Directory-Cert"`. Both of the directory servers must recognize this certificate to perform the comparison.
`-m`	*path*	Specify the path to the security module database. For example, `/usr/iplanet/servers/slapd-`*serverID*`/secmodule.db`.
		You need to specify this option only if the security module database is in a different directory from the certificate database itself.
`-K`	*keyFile*	Specify the file and path name of the client's private key database. This option may be omitted if the key database is in the location already given by the `-P` option.
`-W`	*password*	Specify the password for the client's key database given in the `-K` or `-P` options. This option is required for certificate-based client authentication.

Return Values

The `ldapcmp` tool is based on the Sun ONE LDAP SDK for C and its return values are those of the functions it uses, such as `ldap_simple_bind_s()`, `ldap_search_ext()`, and `ldap_result()`. These functions return both client-side and server-side errors and codes. Table 7-5 shows the possible return values when the directory is a Sun ONE Directory Server. Other LDAP servers may send these values under different circumstances or may send different values. They may also send other result codes entirely; for example, custom result codes from a custom plug-in. For further information about result codes, see the *Sun ONE LDAP SDK for C Programming Guide*.

Table 7-5 Return Values of `ldapcmp`

Return Value	Result Code and Explanation
0 (0x00)	`LDAP_SUCCESS`: the operation was successful.
1 (0x01)	`LDAP_OPERATIONS_ERROR`: sent by Directory Server for general errors encountered by the server when processing the request.
2 (0x02)	`LDAP_PROTOCOL_ERROR`: Directory Server may send this error code in the results for a variety of reasons, such as encountering an error when decoding the BER-encoded request.
3 (0x03)	`LDAP_TIMELIMIT_EXCEEDED`: sent by Directory Server if a comparison search exceeded the default time limit for an operation on the server.
4 (0x04)	`LDAP_SIZELIMIT_EXCEEDED`: sent by Directory Server if a comparison search found more results than the default maximum allowed by the server.
10 (0x0a)	`LDAP_REFERRAL`: sent by Directory Server if the given base DN is an entry not handled by either server and if the referral URL identifies a different server to handle the entry.
11 (0x0b)	`LDAP_ADMINLIMIT_EXCEEDED`: sent by Directory Server if a comparison search found more results than the limit specified by the `lookthroughlimit` directive in the `slapd.conf` configuration file. If not specified in the configuration file, the default limit is 5000.
32 (0x20)	`LDAP_NO_SUCH_OBJECT`: the given base DN cannot be found in both Directory Servers and if no referral URLs are available.
50 (0x32)	`LDAP_INSUFFICIENT_ACCESS`: sent by Directory Server if the DN used for authentication does not have permission to read from the directory.
81 (0x51)	`LDAP_SERVER_DOWN`: either of the LDAP servers did not respond to the comparison search or a connection was lost.
82 (0x52)	`LDAP_LOCAL_ERROR`: an error occurred when receiving the results from either server.
83 (0x53)	`LDAP_ENCODING_ERROR`: the request could not be BER-encoded.
84 (0x54)	`LDAP_DECODING_ERROR`: an error occurred when decoding the BER-encoded results from either server.
89 (0x59)	`LDAP_PARAM_ERROR`: one of the options or parameters is invalid.
90 (0x5a)	`LDAP_NO_MEMORY`: memory cannot be allocated as needed.
91 (0x5b)	`LDAP_CONNECT_ERROR`: a specified hostname or port is invalid.
92 (0x5c)	`LDAP_NOT_SUPPORTED`: the `-V 2` option is needed to access a server that only supports LDAP v2.

Command-Line Examples

The examples in this section demonstrate common uses of the ldapcmp tool. All examples assume the following context:

- All entries in the directories are stored under dc=company,dc=com.
- The directory server has been configured to support anonymous access for search and read. Therefore, you do not have to specify any bind information in order to perform the search.
- The servers are located on the machines named host1 and host2.
- The servers both use port number 389. Because this is the default port, you do not have to specify the port number on the search request.

Comparing Two Directories

By specifying the root DN as the base DN, ldapcmp will search all entries of both directories. The output of the following command will show you *all* differences between the directories' contents:

```
$ ldapcmp -h host1 -h host2 -b "dc=company,dc=com"
```

You should have some idea of the size and differences between your directories before comparing them. Comparing two directories is useful for finding small difference between directories. This output though will be very large and not very helpful if all entries are completely different. The comparison can be narrowed here by specifying the base DN of a similar subtree in both directories.

Comparing Two Entries

The ldapcmp tool can also be used to compare single entries. This operation is much quicker than comparing a subtree because the searches are faster and only a single comparison needs to be performed. The following command uses the DN of the comparative entry as the base DN on the command-line:

```
$ ldapcmp -h host1 -h host2 -s base \
             -b "cn=Pete Minsky,ou=People,dc=company,dc=com"
```

Using LDAP_BASEDN

To simplify the command-line, you can set the base DN using the `LDAP_BASEDN` environment variable. Doing this allows you to avoid specifying the search base with the `-b` option every time you use the `ldapcmp` tool.

NOTE For information on how to set environment variables, see the documentation for your operating environment.

Assuming `LDAP_BASEDN` is set to `dc=company,dc=com.`, the following command will compare your directories on two different hosts:

```
$ ldapcmp -v -h host1 -h host2
```

Specifying the `-v` option for verbose output is helpful because the base DN being used will be displayed in the output for verification.

Comparing Directory Configurations

Directory Server configuration information is stored as entries in the directory itself. You may use the `ldapcmp` tool to compare how each of your servers is configured. The following command-line will compare the root DSE of two Directory Servers:

```
$ ldapcmp -h host1 -h host2 -b ""
```

Because some configuration information is host- and directory-specific, the previous command will always display some differences.

Another source of configuration information is the schema used by your directories. The following command will compare two directory schemas:

```
$ ldapcmp -h host1 -h host2 -b "cn=schema"
```

Schemas can be very large, and comparisons between them are useful only if they are known to have small differences. For example, you can see if a master schema has been customized in different ways for separate directories.

Chapter 8

The LDAPSubtdel Tool

The `LDAPSubtdel` tool provides the means for removing a subtree from a Lightweight Directory Access Protocol (LDAP) directory server. This chapter provides instructions on how to use it. It contains the following sections:

- Overview
- Running LDAPSubtdel
- Options
- Example

Overview

The `LDAPSubtdel` tool enables an administrator to remove a subtree from an LDAP directory server. It opens a connection to an LDAP server, binds, performs a search using a predefined filter, and performs a subtree deletion. It is written in Java™ using the LDAP SDK for Java, therefore requiring a Java runtime environment to function properly. The `LDAPSubtdel` program was compiled with Java 1.2. The latest version of the tool can be found in the *DSRK_base*/`java/LDAPSubtdel` directory. More information on the program can be found in the README in this directory.

Running LDAPSubtdel

The `LDAPSubtdel` tool is delivered as either `ldaplstd.jar` or `lstd.jar`. The difference between how the two versions are executed depends on whether the LDAP SDK is installed. The `ldaplstd.jar` file includes the `ldapjdk.jar`. The `lstd.jar` requires that the LDAP SDK be installed.

With LDAP SDK Installed

To run the tool when the LDAP SDK for Java is installed in the proper directory, execute this command from the directory in which `lstd.jar` is stored:

```
# java -cp ldaplstd.jar:ldap_sdk_dir:ldapjdk.jar LDAPSubtdel
```

where *ldap_sdk_dir* is the directory in which the LDAP SDK is installed.

Without LDAP SDK Installed

To run the tool when the LDAP SDK for Java is not installed, execute this command from the directory in which `ldaplstd.jar` is stored:

```
# java -cp ldaplstd.jar LDAPSubtdel
```

The `ldaplstd.jar` file includes the `ldapjdk.jar`.

NOTE The latest LDAP SDK for Java can be downloaded from Sun Microsystems' Download Center.

Options

Table 8-1 lists the options for the `LDAPSubtdel` tool.

Table 8-1 Options for `LDAPSubtdel`

Option	Parameter	Purpose
`-v`		Verbose output mode: the tool will display additional information about the operations it performs.
`-n`		No-op mode: use with the `-v` option to show what the tool would do with the given input. Does not perform any operations.
`-M`		Manage smart referrals: when they are the target of the update, modify the actual entry containing the referral instead of the entry obtained by following the referral. For more information, see "Creating Smart Referrals" in Chapter 2 of the *Sun ONE Directory Server Administration Guide*.
`-r`		Removes the base DN entry. By default, this entry is kept.
`-d`		Set the LDAP debugging level.

Table 8-1 Options for `LDAPSubtdel`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. When this option is omitted, the default is `localhost`.
`-p`	*port*	Specify the port number for accessing the directory server host. The default is 389 normally and 636 when the SSL options are used.
`-D`	*bindDN*	Specify a bind DN for accessing your directory, usually in double quotes ("") for the shell. If the bind DN and its password are omitted, the tool will use anonymous binding. The bind DN determines what entries and attributes may be modified, according to the DN's access permissions.
`-w`	*password*	Specify the password for the bind DN. **CAUTION: Specifying the password on the command-line is a possible security risk.**
`-V`	*version*	Specify the LDAP protocol version number to be used for the modify operation, either 2 or 3. LDAP v3 is the default; only specify LDAP v2 when connecting to servers that do not support v3.
`-w`	`-`	Type the password for the bind DN when prompted. This is the most secure way of specifying the password.
`-b`	*DN*	Use the DN as the starting point for the search for the subtree to be deleted. It returns no attributes, only DNs.

Example

The example in this section demonstrates a common use of the `LDAPSubtdel` tool. It assumes the directory is on the *localhost* using the default LDAP port and the subtree `ou=People, dc=example,dc=com` will be deleted.

```
$ LDAPsubtdel -v -b dc=example,dc=com \
              -D cn=Directory Manager -w secret -M
```

The exit status is `0` if no errors occur. Errors result in a non-zero exit status with a diagnostic message being written to standard output.

NOTE Included with the distribution of the `LDAPSubtdel` program (in the *DSRK_base*`/java/LDAPSubtdel` directory) is test code, which enables you to test the basic functionality of this program.

Example

Chapter 9

Directory Access Tools Using DSML

The Sun™ ONE Directory Server Resource Kit (DSRK) includes tools that communicate with Sun ONE Directory Server using a Simple Object Access Protocol / Directory Services Markup Language (SOAP / DSML) interface. This chapter provides information on these tools. It contains the following sections:

- Overview
- Usage
- DsmlSearch
- DsmlModify
- Java Naming and Directory Interface

Overview

The LDAP tools using SOAP / DSML are located in the `java/DSML` directory. They include `DsmlSearch` and `DsmlModify`. They are similar to the `ldapsearch` and `ldapmodify` tools (defined in Chapter 3, "The ldapsearch Tool" and Chapter 4, "The ldapmodify Tool," respectively) except they use the SOAP / DSML interface to communicate with Directory Server. A README with information on these tools can also be found in the `java/DSML` directory.

NOTE Directory Services Markup Language (DSML) is a markup language that enables you to represent directory entries and commands in XML. Simple Object Access Protocol (SOAP) is an XML-based protocol used for the exchange of information. Using these two standards directory information can be sent using the HyperText Transfer Protocol (HTTP).

Usage

The DSML tools are written in Java™, therefore requiring a Java runtime environment to function properly. They were compiled using Java 1.3.1, and use Apache SOAP v2.3.1. In order to run properly, the tools need access to the following jar files:

- `soap.jar`
- `xerces.jar`
- `mail.jar`
- `xmisoap.jar`
- `activation.jar`

Ensure that the listed jar files listed are in your `CLASSPATH`.

Tools

The tools included in the DSRK are:

- DsmlSearch
- DsmlModify

They are explained in the following sections.

DsmlSearch

The `DsmlSearch` tool issues search requests to an LDAP directory and displays the result as XML text.

Syntax

The syntax of the `DsmlSearch` tool on the command-line takes the form:

```
java DsmlSearch -h http://host:port -b baseDN [options] filter [attributes...]
```

Where:

- *host:port* are the host name and port number of the directory server.
- *baseDN* defines the base distinguished name (DN) of the search.

- *options* are the command-line options and their parameters described in "DsmlSearch Options."
- *attributes* define the attributes which are being searched.

DsmlSearch Options

Table 9-1 details the options available for the `DsmlSearch` tool.

Table 9-1 Command-line Options for DsmlSearch

Option	Parameter	Purpose
-h	hostURL	Specifies the URL of the directory server.
-b	baseDN	Specifies the base DN of the search.
-D	bindDN	Specifies the bind DN.
-w	passwd	Specifies the bind password (for simple HTTP authentication).
-s	1. scope	1. Specifies the scope of the search.
	2. baseObject	2. For searching only the base entry.
	3. singleLevel	3. For searching only the children.
	4. wholeSubtree	4. For searching the base entry and all children.
-a	1. deref	1. Specifies how aliases are dereferenced.
	2. neverDefAliases	2. Aliases are never dereferenced.
	3. derefFindingBaseObj	3. Dereferenced when finding the base DN.
	4. derefAlways	4. Dereferenced when finding below the base DN.
-l	seconds	Specifies the maximum number of seconds to wait for the search.
-z	number	Specifies the maximum number of entries to return for the search
-f	file	Specifies the name of the file containing the search filter

DsmlModify

The `DsmlModify` tool edits the contents of an LDAP directory, either by adding new entries or by modifying existing ones. The tool takes as input update statements in XML and issues the corresponding request to the designated directory server.

Syntax

The syntax of the `DsmlModify` tool on the command-line takes the form:

```
java DsmlModify -h http://host:port -b baseDN [options] -f Modfile
```

Where:

- *host:port* are the host name and port number of the directory server.
- *baseDN* defines the base DN for the entries to be modified.
- *options* are the command-line options and their parameters described in "DsmlModify Options."
- *Modfile* is the path to the file that contains the entry modifications.

DsmlModify Options

Table 9-2 details the options available for the `DsmlModify` tool.

Table 9-2 Command-line Options for DsmlModify

Option	Parameter	Purpose
`-h`	*hostURL*	Specifies the URL of the directory server.
`-b`	*baseDN*	Specifies base dn to search.
`-D`	*bindDN*	Specifies the bind dn.
`-w`	*passwd*	Specifies the bind password (for simple HTTP authentication).
`-f`	*Modfile*	Specifies the name of a text file that includes the modifications

Java Naming and Directory Interface

Java Naming and Directory Interface (JNDI) is an API specified in Java technology that provides naming and directory functionality to applications written in the Java programming language. The DSRK includes a JNDI service provider for DSML in the *DSRK_base*/`jndi-dsml` directory. More information can be found in Chapter 35, “Java Naming and Directory Interface.”

Part 3

Performance Tools

Chapter 10

The idsktune Optimization Tool

The `idsktune` tool helps automate the process of optimizing server settings to initiate their best performance. This chapter provides instructions on how to use the `idsktune` tool. It contains the following sections:

- Overview
- System Tuning
- Command Usage
- Sample Output

Overview

The default operating system and network settings on many platforms are not suitable for most high performance directory services. Tuning is the process of modifying these settings for optimal performance of both directory clients and directory servers. The `idsktune` tool helps automate this task by checking for necessary patches and suggesting the optimal kernel and TCP/IP settings for running Sun™ ONE Directory Server.

NOTE This tool does not modify the system; it only provides suggested settings. The system administrator must make these changes manually.

`idsktune` is provided with Directory Server in the *DirectoryServer_base*`/bin/slapd/server` directory. However, the DSRK and its updates should include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory.

TIP Run `idsktune` and apply its guidelines before installing Directory Server. Also, in order to get accurate measurements from performance tests, you should use `idsktune` to configure all machines involved in the tests.

System Tuning

The `idsktune` tool gathers information about the operating system, kernel, and TCP stack in order to make tuning recommendations. Because of system differences, most of the tuning parameters in the following sections are for Solaris™ and UNIX® systems only. On the Microsoft® Windows platforms, the `idsktune` tool verifies operating system compatibility and service patch level.

OS and Kernel Settings

The tool displays current operating system version numbers and patch information, along with any recommended patches. It also verifies disk and memory availability and warns of any deficiencies. Specifically, `idsktune` verifies and reports on the following:

- Up-to-date operating system and kernel versions:
  - Solaris and Red Hat Linux version numbers
  - Solaris kernel build date
  - Solaris, HP-UX, and AIX patches
- Sufficient memory and disk space:
  - Physical memory size
  - Swap space size or swap partition size
  - Memory resource limits
  - File descriptor resource limits
- Scheduler settings:
  - Maximum threads per process (HP-UX)
  - Maximum files (HP-UX)

TCP Settings

The idsktune tool reads the current settings of your system's Transmission Control Protocol (TCP) module and makes recommendations for changes. The tool does not perform any of these modifications. Instead, it displays the command-line for the ndd tool, which the system administrator should use to set the parameter. The ndd tool is available on the Solaris platform. An equivalent tool should be used if you are tuning a different platform.

NOTE The system administrator should consider local network conditions and other application needs when modifying TCP settings. In general, however, the recommendations will optimize performance whether the machine is dedicated to the directory server or shared with other applications.

The idsktune tool verifies and makes recommendations on the following settings:

- Listen backlog queue size
- tcbhashsize,tcbhashnum and tcp_msl
- sominconn and somaxconn
- ipport_userreserved_min
- tcp_close_wait_interval and tcp_time_wait_interval
- tcp_keepalive_interval
- tcp_max_listen
- tcp_conn_request_max
- tcp_conn_req_max_q and tcp_conn_req_max_q0
- tcp_rexmit_interval_initial
- net.inet.ip.portrange.hifirst (Linux) and tcp_smallest_anon_port
- tcp_slow_start_initial
- net.inet.tcp.delayed_ack (Linux) and tcp_deferred_ack_interval
- link_speed on /dev/hme (Solaris platform)

The idsktune tool verifies all of the settings for the Solaris 2.6 and Solaris 8 operating systems that are described in "Tuning System Settings" in Chapter 5 of the *Sun ONE Directory Server Installation and Tuning Guide.*

Further Information

Basic and advanced information about tuning your Solaris system is available through the following books and websites:

- *Sun Performance and Tuning: Java and the Internet* (ISBN 0-13-095249-4)
- *Solaris Performance Administration* (ISBN 0-07-011768-3)
- "Solaris 2.x - Tuning Your TCP / IP Stack and More" (`http://www.sean.de/Solaris/soltune.html`)
- "Solaris Tunable Parameters Reference Manual" (`http://docs.sun.com/doc/817-1759`)
- "Tuning Tru64 UNIX for Internet Servers" (`http://h30097.www3.hp.com/docs/internet/TITLE.HTM`)
- `sys-check` tool for Tru64 (OSF / 1) UNIX (`http://h30097.www3.hp.com/sys_check/resources.html`)

Command Usage

Although this command should be run as `root` to get a full report on all settings, most settings are verified when it is run as any user. However, you must be `root` if you wish to modify any of the operating system, kernel, or TCP settings based on the `idsktune` recommendations.

Syntax

The syntax of the `idsktune` command takes the following form:

```
idsktune [-v | -D | -q | -c | -\?]
```

Options

Table 10-1 details the command-line options for `idsktune`.

Table 10-1 Command-Line Options for `idsktune`

Option	Parameter	Purpose
`-v`		Version: gives the build date identifying the version of the tool.

Table 10-1 Command-Line Options for `idsktune` *(Continued)*

Option	Parameter	Purpose
`-D`		Debug mode: the output includes the commands it runs internally, preceded by the "`DEBUG`" heading.
`-q`		Quiet mode: the output includes tuning recommendations, but OS version statements are omitted.
`-c`		Client-specific tuning: the output includes tuning recommendations for running a directory client application, but server-specific recommendations are omitted.
`-\?`		Display the usage help text that briefly describes all options.

Sample Output

Code Example 10-1 is an example of running `idsktune` on an untuned Solaris 8 system. This is the default output displayed when no options are used.

Code Example 10-1 Sample Output of `idsktune`

```
# /opt/iPlanet/bin/dsrk52/idsktune
iPlanet Directory Server system tuning analysis
version 11-JAN-2002.
Copyright 2001 Sun Microsystems, Inc.

NOTICE : System is usparc-sun-solaris5.8 (SUNW,Ultra-5_10)
(1 processor).

NOTICE : Patch 109137-01 is not installed.

NOTICE : Patch 109320-01 is not installed.

NOTICE : Patch 108974-02 is not installed.

NOTICE : Patch 108977-01 is not installed.

NOTICE : Patch 108968-02 is not installed.

NOTICE : Patch 108975-02 is not installed.

NOTICE : Patch 108528-01 is not installed.

NOTICE : Patch 108875-07 is not installed.

NOTICE : Patch 108652-13 is not installed.

NOTICE : Solaris patches can be obtained from
http://sunsolve.sun.com or your Solaris support representative.
```

Code Example 10-1 Sample Output of idsktune *(Continued)*

```
ERROR  : Only 128MB of physical memory is available on the system.
1024MB is the recommended minimum.

ERROR  : There is 128MB of physical memory but only 114MB of swap space.

WARNING: The tcp_close_wait_interval is set to 240000 milli-
seconds (240 seconds). This value should be reduced to allow for
more simultaneous connections to the server.  A line similar to
the following should be added to the /etc/init.d/inetinit file:
ndd -set /dev/tcp tcp_time_wait_interval 30000

NOTICE : The tcp_conn_req_max_q value is currently 128, which will limit the
value of listen backlog which can be configured. It can be raised by adding
to /etc/init.d/inetinit, after any adb command, a line similar to:
ndd -set /dev/tcp tcp_conn_req_max_q 1024

NOTICE : The tcp_keepalive_interval is set to 7200000 milli-seconds (120
minutes). This may cause temporary server congestion from lost client
connections.

NOTICE : The tcp_keepalive_interval can be reduced by adding the
following line to /etc/init.d/inetinit:
ndd -set /dev/tcp tcp_keepalive_interval 600000

NOTICE : The NDD tcp_rexmit_interval_initial is currently set to
3000 milliseconds (3 seconds). This may cause packet loss for
clients on Solaris 2.5.1 due to a bug in that version of Solaris.
If clients are not using Solaris 2.5.1, no problems should occur.

NOTICE : If the directory is service is intended only for LAN or private
high-speed WAN environment, this interval can be reduced by adding to
/etc/init.d/inetinit: ndd -set /dev/tcp tcp_rexmit_interval_initial 500

NOTICE : The NDD tcp_smallest_anon_port is currently 32768. This allows a
maximum of 32768 simultaneous connections. More ports can be made available
by adding a line to /etc/init.d/inetinit:
ndd -set /dev/tcp tcp_smallest_anon_port 8192

WARNING: tcp_deferred_ack_interval is currently 100 milliseconds. This will
cause Solaris to insert artificial delays in the LDAP protocol. It should be
reduced during load testing. This line can be added to the
/etc/init.d/inetinit file:
ndd -set /dev/tcp tcp_deferred_ack_interval 5

WARNING: There are only 1024 file descriptors available, which limits the
number of simultaneous connections. Additional file descriptors, up to
65536, are available by adding to /etc/system a line such as:
set rlim_fd_max=4096

NOTICE : / partition has less space available, 213MB, than the
largest allowable core file size of 242MB. A daemon process which
dumps core could cause the root partition to be filled.

ERROR  : The above errors MUST be corrected before proceeding.
```

Chapter 11

The ldclt Stress Test Tool

The `ldclt` tool is a powerful and flexible client for creating and running stress tests on Lightweight Directory Access Protocol (LDAP) directory servers. The tool generates LDIF files and performs searches, additions, modifications, deletions, or renaming using complex randomizing capabilities to simulate real usage conditions. This chapter provides instructions on how to use the `ldclt` tool. It contains the following sections:

- Overview
- Command Usage
- Error Handling
- Sample Output
- Command-Line Examples
- Troubleshooting

Overview

The `ldclt` (LDAP client) is a multithreaded application developed using the Sun™ ONE LDAP SDK for C. It is designed to be used in automated tests on a directory; its command-line interface is extensive and suitable for scripting. This will allow you to run complex tests automatically and compare performance over time for reliability. The DSRK and its updates include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory.

NOTE To insure that scripts can be easily maintained, the `ldclt` tool continues to provide a high degree of backward compatibility.

ldclt uses a minimum of 8 MB of memory and may use much more depending upon the chosen options and operations. If necessary, it sets its own resource limits according to the options and data it processes from the directory.

NOTE See the man pages for ulimit and getrlimit for more information.

The ldclt tool continuously displays its performance measurements on the standard output. By default, ldclt will display performance statistics cumulated over all threads every 10 seconds. And every 15 minutes (and also upon exit), the tool will display a cumulative count of operations performed (global statistics) and errors encountered. (See the -e noglobalstats option as detailed in "General Options" on page 137 to suppress the global statistics output.) Several error management options allow the tool to handle most errors without interrupting the test. Fatal errors will be reflected in the tool's exit status.

Command Usage

The ldclt tool performs repeated operations on a designated directory server and measures the average completion time for requests. The tool is multithreaded and can be configured to perform high load testing. All threads loop continuously to perform the same LDAP operation repeatedly but with different input values each time. The possible operations are binds, searches, additions, modifications (including RDNs whose modifications are also referred to as *reparenting*), and deletions.

Syntax

The syntax of the ldclt tool on the command-line takes the following form:

```
ldclt [options] -b "baseDN" [-f patternString | -e rdn=patternString,...] \
      [-e bindonly|esearch|add|attreplace|delete|rename|genldif]
```

Where:

- *options* are the numerous command-line options and their parameters described in "Options" on page 137.
- *baseDN* is the base of the search or the suffix used for other operations, enclosed in double quotes ("") for the shell.

- *patternString* is a filter or RDN that may include variants for performing randomization substitutions.
- The `-e` option of the syntax defines the minimal options for performing other LDAP operations, respectively: binding, searches, additions, modifications, deletions, reparenting, and generating LDIF files. See the Tip on page 137 for more information.

NOTE The `ldclt` tool is highly configurable through its command-line options, and it is rarely used in the simple forms presented here. See “Sample Output” on page 147, for an explanation of the options for controlling each feature and their possible combinations.

Options

The many `ldclt` tool options allow you to randomize the values of most LDAP operation parameters in order to simulate realistic client requests. They also allow you to control how often, and for how long, multiple threads will run these operations, simulating a given load on your directory server. The `ldclt` tool has four types of options:

- General Options
- Bind Options
- Target Object Options
- Operation-specific Options

The following sections detail these options. The `ldclt -H`, `-h`, or `-?` options when run on the command-line will display brief descriptions of all the command-line options.

TIP The `ldclt` tool has a special `-e` option for specifying multiple execution parameters. The `-e` option appears a number of times in the following tables to describe these parameters separately. When using the `-e` option on the command-line, you may specify its parameters in a comma separated list, for example:

```
ldclt -e noglobalstats,delayedstartup=10,counteach
```

General Options

The general options listed in Table 11-1 control multi-threading, execution time, error handling and output.

Table 11-1 General Options for `ldclt`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname or IP address of the directory server. When this option is omitted, the default is the local host. The `ldclt` tool supports full IPv6 addressing.
`-p`	*port number*	Specify the port number for accessing the directory server host. The default is 389 when this option is omitted.
`-n`	*number of threads*	Specify the number of threads that will perform LDAP operations in parallel. The output displays the average performance of all threads combined. The default number of threads is 10. The maximum number is 4000, although most systems are limited to less by the maximum threads and file descriptors per process.
`-e`	*number of seconds*	Specify the delay time in seconds between the startup of each thread. By default, all threads start at the same time. With this parameter, each thread will sleep for *threadNumber* x *seconds* before starting.
`-a`	*number of concurrent operations*	Run the tool in asynchronous operation mode, with each thread having at most *maxPending* concurrent operations. A pending operation is one whose results have not yet been read from the server. Each thread will also maintain a minimum of *maxPending* / 2 concurrent operations.
`-N`	*number of intervals*	Specify the total number of intervals, and therefore the total time, for which the tool should run. For example, specify `-N 360` to run for one hour with the 10 second intervals. By default, there is no limit to the number of intervals, and the tool will run indefinitely.
`-T`	*number of operations*	Specify the total number of operations, cumulated across all threads, to perform before exiting. The tool will exit after the interval where this number of operations is reached or exceeded, thus more than the total operations may be performed. By default, there is no limit to the number of operations, and the tool will run indefinitely.
`-W`	*number of seconds*	Specify the waiting time, in seconds, between each operation of each thread. By default there is no waiting time (0 seconds).
`-t`	*number of seconds*	Specify the server-side time-out period, in seconds, for all LDAP operations performed. When this option is omitted, the default is 30 seconds.
`-e`	`dontsleeponserverdown`	By default, an `ldclt` thread will sleep 1 second if it receives an error indicating that the server is down (code 81 or 91). This parameter will prevent threads from sleeping, allowing them to loop and repeatedly attempt operations. Due to the built-in resource management, the `ldclt` tool may use 100% of all CPU time and block the client machine when using this parameter.

Table 11-1 General Options for `ldclt` *(Continued)*

Option	Parameter	Purpose
`-i`	*number of intervals*	Specify the number of inactive intervals for a thread before a warning message is sent. A thread is inactive or starving if, for whatever reason, it performs no operation during an interval. By default, a warning message is sent on the interval after 3 inactive intervals (40 seconds total).
`-E`	*number of allowed error occurrences*	Specify the maximum number of allowed occurrences of any one error, and exit after the interval when that number is reached or exceeded. When this option is omitted, the default is 1000. This option only applies to errors not ignored by the `-I` option.
`-I`	*numerical error code*	Specify a numerical error code to ignore when returned by an operation. This option can be used multiple times, once for each error to ignore. See “Error Handling” on page 146 for more information.
`-e`	`counteach`	Specify that the statistics include all operations attempted. For example, even a search that failed because a random value was not found will be reported. By default, only operations that succeed are counted. With this option, the output will more accurately represent thread activity and server response. The following errors will be counted: • When performing searches: LDAP_NO_SUCH_OBJECT (code 32) • When performing additions: LDAP_ALREADY_EXISTS (code 68) • When performing deletions: LDAP_NO_SUCH_OBJECT (code 32) • When renaming entries: LDAP_ALREADY_EXISTS (code 68) or LDAP_NO_SUCH_OBJECT (code 32) or LDAP_PROTOCOL_ERROR (code 2)
`-e`	`v2`	Perform all operations using LDAP v2 protocol version. LDAP v3 is the default; specify LDAP v2 only when connecting to servers that do not support v3.
`-e`	`noglobalstats`	Suppress the periodical global statistics output. By default, the global statistics will be printed every 90 intervals. See “Sample Output” on page 147 for more information.
`-v`		(Small v) Specify a verbose output, which provides additional information about the operations that `ldclt` is performing. This option may be used in conjunction with `-q` to display a useful report of tool activity without too many error messages.
`-V`		(Capital V) Specify a VERY verbose output, which shows debugging information about the operations that `ldclt` is performing. Specify this option with `-v`, but not `-q` or `-Q`, to display all possible output.
`-q`		(Small q) Specify a quiet output. None of the bind errors will be reported, and intermediate node creation messages are not displayed. Use with the `-I` option to suppress output of ignored errors.

Table 11-1 General Options for `ldclt` *(Continued)*

Option	Parameter	Purpose
`-Q`		(Capital Q) Specify a VERY quiet output. Bind errors are not reported, intermediate node creation messages are not displayed, and all messages regarding thread activity and asynchronous, pending requests are suppressed.
`-H`		Display the usage help text that briefly describes all options.

Bind Options

The options in Table 11-2 determine how the `ldclt` tool will bind to the directory server and possibly establish a secure connection.

Table 11-2 Bind Options for `ldclt`

Option	Parameter	Purpose
`-D`	*bind DN*	Specify a bind DN for accessing your directory, usually in double quotes ("") for the shell. The bind DN should have sufficient access permissions for the test operations you wish to perform. If the bind DN and password are omitted, the tool will use anonymous authentication: it will not bind before performing operations. Usually, only search operations may be performed anonymously. To use a different bind DN with every operation, use X placeholders in the *bindDN* and specify the `-e randombinddn` options. For more information, see “Random Binding” on page 150.
`-w`	*password*	Specify the password for the bind DN. The *password* may also include X placeholders to include the same randomly generated value that is substituted into the corresponding bind DN.
`-e randombinddn`		Use this option to replace X placeholders in the bind DN and password options with a random value. You must specify a range of random numbers with the `randombinddnlow` and `randombinddnhigh` options detailed in the following table cells.
`-e randombinddnlow=`*integer*		Specify the low end of the possible range for random integers to be substituted into the bind DN and password options.
`-e randombinddnhigh=`*integer*		Specify the high end of the possible range for random integers to be substituted into the bind DN and password options.
`-e randombinddnfromfile=`*filename*		Specify the full path and filename to a file that contains bind DN and password pairs. The bind credentials will be randomly selected from this file each time the tool binds to the server. (See “Bind Operations” on page 149 for more information.) Do not use `-D`, `-w`, and `-e randombinddn` with this option.

Table 11-2 Bind Options for `ldclt` *(Continued)*

Option	Parameter	Purpose
`-e`	`referral=on|off|rebind`	Specify the binding policy when following referrals: • on - Follow referrals using anonymous binding; this is the default when this option is omitted. • `off` - Do not follow referrals. • `rebind` - Follow referrals by rebinding with the same credentials.
`-e`	`smoothshutdown`	Force each thread to finish the operation in progress and unbind before terminating for whatever reason. With this option, the tool will wait up to 20 seconds for operations to finish. Without this option, pending operations and connections will be dropped upon any forced exit.
`-e`	`bindeach`	Specify that the tool will rebind for each operation it performs. When using this option, the performance measurements in the output include the time required to establish the connection, bind to the directory and perform the given operation. This behavior simulates many clients performing quick operations.
`-e`	`close`	Specify how the tool will disconnect from the server. With this option, the tool will simply close the connection without unbinding. By default, the tool will perform the LDAP unbind operation. This option is often used with `-e bindeach` to simulate connections being dropped by clients.
`-Z`	*path to certificate database file directory*	Establish secure communications using SSL (Secure Socket Layer) client authentication for all operations. This allows you to test the performance of your server over SSL. The *certPath* parameter specifies the directory containing the `cert7.db` certificate database file. (See “SSL Authentication” on page 151.) You must also specify the bind DN and password options, as well as the following three parameters.
`-e`	`cltcertname=`*certificate name*	Specify the name of the certificate to be found in the directory given by the `-Z` option. This certificate will be used to identify the client when establishing an SSL connection with the server.
`-e`	`keydbfile=`*keyDBfilename*	Specify the full path and filename of the key database on the client machine. This file should only be used for testing because the corresponding password is exposed on the command-line.
`-e`	`keydbpin=`*key database password*	Specify the password needed to access the key database file given in the previous option. **CAUTION: This option exposes the password to the SSL key database on the command-line. You should use a special certificate and key database, used only for testing purposes, to avoid revealing your actual password.**

Target Object Options

The options in Table 11-3 determine the target object, either its base DN, its RDN (relative DN), its attribute values, or any combination thereof. Use these options to specify either specific or random entries and create random attribute values, depending upon the type of operation being performed. For example, you may want to add 100 sequentially-named entries with random contents or delete random entries (RDNs) under a given base DN. See "Sample Output" on page 147, for more information.

Table 11-3 Target Object Options for `ldclt`

Option	Parameter	Purpose
`-b`	*base DN*	Specify the base DN or suffix for all operations that require one, usually in double quotes ("") for the shell. To use a different base DN with every operation, use X placeholders in this option and specify the following `-e randombase` option. For more information see "Random Base DNs" on page 156.
`-e randombase`		Use this option to replace X placeholders in the *base DN* with random values. You must specify a range of random numbers with the `randombaselow` and `randombasehigh` options detailed in the following two table cells.
`-e randombase low=`*integer*		Specify the low end of the possible range for random integers to be substituted into the *base DN* option.
`-e randombase high=`*integer*		Specify the high end of the possible range for random integers to be substituted into the *base DN* option.
`-f`	*patternString*	Specify a filter string for search operations or the RDN pattern for other operations. The *patternString* has the format *attribute=value*, and the *value* may contain X placeholders for any of the substitution features described by the following options. See "Random Filters and RDNs" on page 152 for further details. **CAUTION: This option is mutually exclusive with `-e rdn=`*patternString*.**
`-e incr[,noloop]`		Activate the incremental counter feature to replace X placeholders with sequential integers. Use with the `-r` and `-R` options to specify the starting and ending values of the counter, respectively. Specify the `noloop` option to count the range only once. In this case, the corresponding thread will die upon reaching the upper bound.
`-e commoncounter`		Specify that all threads use and increment the same thread-safe counters. This option applies to all counters in use, such as those for *patternString* substitutions and those for template substitutions.
`-e random`		Activate the random value substitution feature to replace X placeholders. Specify the `-r` and `-R` options to define the range of random integers for substitution. Alternatively, use with the `-e string[,ascii]` options to substitute random strings.

Table 11-3 Target Object Options for `ldclt` *(Continued)*

Option	Parameter	Purpose
`-r`	*integer*	Specify the lower bound of the incremental counter or random value range.
`-R`	*integer*	Specify the upper bound of the incremental counter or random value range.
`-e`	`string[,ascii]`	Activate the random string substitution feature to replace `X` placeholders. This option requires the `-e random` option. By default, UTF-8 characters will replace each of the `X` placeholders. Use the `ascii` option to generate only 7-bit characters (hexadecimal values less than or equal to `0x7f`).
`-e`	`rdn=`*'patternString'*	The *patternString* may specify either a filter pattern for search operations or an RDN pattern for other operations. The *patternString* has the format *attribute:value*, and instead of using `X` placeholders, the *value* uses a special syntax described in "Random Filters and RDNs" on page 152. Use single quotes (`''`) to use this special syntax with the shell. This option is mutually exclusive with the `-f`, `-e incr[,noloop]`, `-e random`, `-r`, `-R`, and `-e string[,ascii]` options. However, it requires the `-e object=`*filename* option with a valid template file, even if this file is not used to generate entries.
`-e`	`object=`*filename*	Specify the path and name of a file containing an entry template for generating random entries (see "Random Entry Generation" on page 157). Requires the `-e rdn=`*patternString* option.

Operation-specific Options

The options in Table 11-4 are operation-specific options. All of the options for specifying operations, namely `-e bindonly`, `-e esearch`, `-e add`, `-e attreplace`, `-e delete`, `-e rename`, and `-e genldif` are mutually exclusive. See the corresponding section under "Sample Output" on page 147, for more information about each operation.

Table 11-4 Operation-Specific Options for `ldclt`

Option	Parameter	Purpose
`-e`	`bindonly`	Specify that only bind and unbind operations be performed repeatedly, with no other intervening LDAP operations. The `-e close` option will simulate client disconnects without unbinding. See "Bind Operations" on page 149 for more information.
`-e`	`esearch`	Perform searches with `-f` *patternString* or `-e rdn=`*patternString* as the filter. In this release of the tool, searches with large results are very inefficient and should be avoided. Use wildcards or operators other than equality only if they return few entries. This restriction will be fixed in future versions. See "Searches" on page 151 for more information.

Table 11-4 Operation-Specific Options for `ldclt` *(Continued)*

Option	Parameter	Purpose
`-s`	*scope*	Specify the scope of search operations. The *scope* parameter may be one of the following: 1. `base` - Search only the base entry itself. 2. `one` - Search only the children of the base entry. 3. `subtree` - Search the base entry and all its descendants. This is the default if the `-s` option is omitted.
`-e`	`attrlist=`*attribute[:attribute]...*	Specify the type names of the attributes to retrieve for entries in the search results. When this option is omitted, all attributes will be retrieved by default.
`-e`	`randomattrlist=`*attribute[:attribute]...*	Specify the type names of possible attributes to retrieve for entries in the search results. For every search, the tool will randomly select one of these attributes to be retrieved for all entries in the results.
`-e`	`attrsonly=0|1`	Specify whether attribute values are returned (`0`) or not (`1`), along with the attribute names of the entries in the search results. The `ldclt` tool will use less memory when attribute values are not returned. By default, attribute values are returned (`0`).
`-e`	`add`	Perform LDAP add operations, using DNs determined by the base DN and by the RDN *patternString* of either the `-f` or `-e rdn` options. You must also specify the one of the following object class template options or create a custom template in a file given by the `-e object` parameter. See “Adding Entries” on page 156 for more information.
`-e`	`person|emailperson|inetorgperson`	Specify one of the predefined object class templates for adding or generating entries. See “Random Entry Generation” on page 157 for more information. Both `emailperson` and `inetorperson` require the following `imagesdir` option.
`-e`	`imagesdir=`*path*	Specify the full path of the directory containing JPEG image files. Only files with the `.jpg` extension in this directory will be used. The `ldclt` tool will take the files sequentially from this directory when generating `emailperson` and `inetorperson` entries. The files are interpreted as binary image files but their actual contents is unimportant. Files may be as large as needed to test your directory.
`-e`	`genldif=`*filename*`[,append]`	Specify the full path and name of a file to create with the generated entries in LDIF format. This operation is similar to an addition and uses the same options, except the new entries are written to the file (see “Random Entry Generation” on page 157). Unless you specify the append option, the tool will exit with an error if the file exists.
`-e`	`attreplace=`*attribute*`:`*value*	Perform LDAP modify operations on entries determined by the base DN and the RDN of the `-f` *patternString* option. Specify the *attribute* to modify and a *value* string containing `X` placeholders: the tool will replace each placeholder with a random character with every operation (no other options are needed). See “Modify Operations” on page 161.

Table 11-4 Operation-Specific Options for `ldclt` *(Continued)*

Option	Parameter	Purpose
`-e`	`delete`	Perform LDAP delete operations on entries determined by the base DN and the RDN *patternString* of either the `-f` or `-e rdn` options. See "Delete Operations" on page 162 for more details.
`-e`	`rename[,withnewparent]`	Perform LDAP modRDN operations on entries determined by the base DN and the RDN *patternString* of either the `-f` or `-e rdn` options. The new RDN will be determined from the same *patternString* which therefore cannot rely on incremental value substitutions. Specify the `withnewparent` option to simulate a reparent operation if you are also using the random base DN options. See "modRDN Operations" on page 162 for more details.

Exit Status

By default, the `ldclt` tool will run indefinitely, performing operations repeatedly. It will only exit in one of the following conditions:

- There is a usage error, a fatal initialization error, or a fatal runtime error.
- Any one of the LDAP errors not being ignored reaches the occurrence limit.
- All threads terminate or die.
- The interval limit specified by the `-N` option or the operation limit specified by the -T option is reached.
- The tool is interrupted by `SIGINT` (`^C` or `kill -2`).

Upon termination, the `ldclt` tool will return one of the following values to the shell. When possible, the display will provide further information about any errors.

Table 11-5 Exit Status of `ldclt`

Exit Status	Explanation
`0`	No problem during execution.
`1`	Serious error that should not occur (please report to `idsrk@Sun.COM`).
`2`	Error in the parameter usage.
`3`	Maximum limit of LDAP errors reached (see "Exit Status 3" on page 164).
`4`	Unable to bind to the directory server.
`5`	Cannot load `libssl` for secure binding.
`6`	Multithreading error (mutex problem).

Table 11-5 Exit Status of `ldclt` *(Continued)*

Exit Status	Explanation
7	Initialization problem, for example, if the tool cannot open or create a file.
8	Resource limitation such as a `malloc` error for memory.
99	Unknown error (please report to `idsrk@Sun.COM`).

Error Handling

The `ldclt` tool handles errors that occur as a result of LDAP operations. Each thread handles errors individually: for example, a connection error will cause only that thread to die. Some errors are expected when performing hundreds and thousands of operations with randomly generated parameters. Operational errors are not fatal, and each thread will continue performing operations with new parameters.

TIP See “Troubleshooting” on page 163 for more information about specific errors and how to avoid them.

Allowing Error Occurences

The tool will count the occurrences of each error code, and by default, it will terminate only if any one error code occurs more than 1000 times. This behavior allows occasional errors but detects server conditions or test configurations that will probably not give valid results. You may use the `-E` option to set a different occurrence limit, for example, to detect all errors when developing tests. See Table 11-1 on page 138 for more information.

Ignoring Error Occurences

If you always expect a given error to occur, use the `-I` option to ignore it in all threads. This error will be exempt from the error occurrence limit and it will not be displayed if the `-q` option is specified. For example, specify `-I 32` (`LDAP_NO_SUCH_OBJECT`) when performing random delete operations or -I 68 (`LDAP_ALREADY_EXISTS`) when performing random add operations.

Normally, connection errors are fatal to the thread in which they occur. The `-I` option may be used to ignore connection errors 81 (`LDAP_SERVER_DOWN`) and 91 (`LDAP_CONNECT_ERROR`) when they occur during a bind. In this case the thread will attempt to rebind instead of exiting. However, a connection error that occurs during an operation is always fatal.

NOTE This tool will trap the following signals:

- `SIGINT` (^C or `kill -2`): display the global statistics report and exit the tool normally.
- `SIGQUIT` (^\ or `kill -3`): display the current statistics but do not exit.

Sample Output

Code Example 11-1 is a sample output when using 20 threads.

Code Example 11-1 Sample Display Using Twenty Threads

```
Average rate:    3.75/thr  (   7.50/sec), total:     75
Average rate:    3.75/thr  (   7.50/sec), total:     75
Average rate:    3.60/thr  (   7.20/sec), total:     72
Average rate:    3.50/thr  (   7.00/sec), total:     70
Global average rate: 9221.70/thr  ( 10.25/sec), total:  92217
Global number times "no activity" reports: never
Global error 19 (Constraint violation) occurs   377 times
Global error 34 (Invalid DN syntax) occurs     3 times
Global error 68 (Already exists) occurs  2772 times
Average rate:    3.90/thr  (   7.80/sec), total:     78
Average rate:    3.80/thr  (   7.60/sec), total:     76
Average rate:    3.65/thr  (   7.30/sec), total:     73
Average rate:    3.75/thr  (   7.50/sec), total:     75
```

Performance Statistics

The average operation rates in Code Example 11-1 provide the following information (from right to left):

- The total number of successful operations during the elapsed interval, all threads considered. In the first line of the sample, the number of successful operations is 75.

- The average number of operations per second during the elapsed interval, all threads considered. In the first line of the sample, the average number of operations per second during the elapsed interval is `7.5`.
- The average rate or number of operations per thread during the entire interval. In the first line of the sample, the average number of operations per second during the entire interval is `3.75`.

By default, the statistics only take into account successful operations. This may misrepresent client and server activity when performing operations that may fail naturally; for example, when performing delete operations with random relative distinguished names (RDNs). Use the `-e counteach` option as detailed in "General Options" on page 137 to count all operations in the statistics. In the asynchronous mode (the `-a` option also detailed in "General Options" on page 137), all operations are counted by default.

Global Statistics

The global statistics give the cumulated operation rates and the error counts for the entire elapsed runtime of the tool. In addition, when running the tool in asynchronous mode (again, the `-a` option detailed in "General Options" on page 137), the global statistics will include the status of pending operations for all threads. The global statistics detailed in Code Example 11-1 provide the following information:

- Comparison of the global average number of operations per second with that of previous intervals shows whether directory performance is improving or degrading over the time of the test.
- The number of "`no activity`" reports tells you if threads have faced starvation problems. By default, the tool expects each thread to perform at least one operation every 40 seconds. (See the `-i` option detailed in "General Options" on page 137.) Threads may be unable to perform operations when other operations are running too long, thus indicating a possible performance problem.

NOTE The output may also include the number of threads that have died, indicating either the end of a non-looping counter or a connection problem

- The global error reports show how many times LDAP errors have been reported. It is natural for some LDAP errors to occur during normal operation, mostly due to the randomization features. However, you may also detect abnormal errors here. See “Troubleshooting” on page 163 for an explanation of the most common errors.

You may request the global statistics at any time using the `SIGQUIT` signal, available on UNIX® systems via the `^\` (Ctrl-Backslash) key sequence. However, this may interrupt the LDAP operation in one or more threads and impact the current interval statistics.

NOTE The `ldclt` statistics are not absolute measurements due to the counting mechanism and the overhead of interval timing. Measurements are more accurate for faster machines. However, unless the `ldclt` tool is running on an overloaded machine, it can submit more operations than a server can process. Therefore, it provides accurate overall results about the number of operations per second.

Command-Line Examples

The large number of options for the `ldclt` tool allows you to configure the behavior for many advanced features. This section explains some of these features and how the options interact to implement the desired behavior. Many of these examples show how to randomize values such as the bind DN or the target DN with each operation. Randomization simulates realistic usage of a directory server in performance tests. You should adjust the command-line parameters so that the range of randomization values most closely resembles your expected directory usage. Other examples of `ldclt` usage can be found in *DSRK_base*`/examples/ldclt`.

NOTE Due to requirements for backwards compatibility, the `ldclt` tool has several mechanisms for generating randomized values.

Bind Operations

The following sections describe how to use these options detailed in Table 11-2 on page 140 to randomize the bind credentials or to use secure binding.

Random Binding

Random binding uses different bind DN and password credentials to perform simple authentication with every LDAP bind operation. In order to use a different, numeric bind DN with every operation, you must specify the `-e randombinddn` options and use `X` character placeholders in the `-D` *bindDN* and `-w` *password* parameters. The same random value will be substituted into both the placeholders (for example, cn=m123 and passwd123). Random binding is specified with the `-e bindeach` option.

CAUTION The capital letter X cannot be escaped for normal usage.

In the following example, each thread of the command will bind and unbind (`-e bindeach,bindonly`) using any one of the 100,000 possible bind credentials:

```
$ ldclt -h host -p port -e bindeach,bindonly \
        -D "uid=XXXXX,dc=example,dc=com" -w testXXXXX \
        -e randombinddn,randombinddnlow=0,randombinddnhigh=99999
```

The values substituted into the bind DN are generated independently of other random substitutions into base DNs, filters, or RDNs. See “Random Filters and RDNs” on page 152 for more information.

CAUTION When using random bind credentials, your test directory should be initialized with entries corresponding to the possible bind DNs and passwords that can be generated. Using random strings in the bind credentials may result in bind errors if a DN does not exist. Specify the `-I 49` option to make the tool ignore `error 49`, `LDAP_INVALID_CREDENTIALS`. A test that uses this option will measure a mix of bind successes and failures. See “Adding Entries” on page 156 for `ldclt` examples that will add all entries in a sequence of RDNs.

Alternatively, specify the `-e randombindfromfile` option to use bind credentials defined in a text file. The file must contain one bind DN and password pair per line, separated by any number of tabs. Leading and trailing space characters are significant in both the DN and password. As shown in the following example, the `-D` and `-w` options are not needed in this case:

```
$ ldclt -h host -p port -e bindeach,bindonly \
        -e randombinddnfromfile=/home/user/testbind.txt
```

Using random bind credentials from a file allows you to write or generate DNs and passwords that test special conditions in your directory. For example, you might want to test search operations involving special ACIs (access control instructions) that apply to the DNs specified in the file.

SSL Authentication

The `ldclt` tool supports client authentication over SSL in order to perform tests using secure binding. The following example shows the options required:

```
$ ldclt -h host -p port -e bindeach,bindonly -Z certPath \
        -e cltcertname=certName,keydbfile=fileName,keydbpin=password
```

The *certPath* and *certName* determine the certificate that will identify the client, and the *fileName* and *password* provide access to the client's keys. Because the password to the key database is exposed on the command-line, we recommend using a special key database that is only used for testing purposes.

CAUTION No randomization is possible with client authentication over SSL, therefore all threads will bind with the same certificate every time. However, you can still use the `-e bindeach` option to measure SSL binding and unbinding with every operation.

Searches

The numerous options for performing searches allow you to randomize what DN is being sought, where to look for it, and what to retrieve upon finding it. With many threads performing many different searches, the `ldclt` tool can simulate a realistic load on your directory server and measure performance under deployment-like conditions. However, `ldclt` does not verify or display any search results, which are discarded immediately.

Anonymous Searches

If you do not specify any bind credentials (as shown in the following command), searches may be performed anonymously:

```
$ ldclt -h host -p port -b "ou=people,dc=example,dc=com" \
        -f uid=XXXXX -e esearch,random -r0 -R99999 \
        -I 32 -e randomattrlist=cn:sn:uid:ou:l:fax:mobile
```

This example performs searches for a random UID in entries below the "ou=people,dc=example,dc=com" base DN. Base DNs may be randomized in searches, using the options described in “Random Base DNs” on page 156. This example also uses a randomized filter string to perform a different search and retrieve a different entry for every search. The syntax for this feature is described in “Random Filters and RDNs” on page 152. The `-I 32` option will ignore code 32 (LDAP_NO_SUCH_OBJECT) errors that occur often with random searches. If the search is successful, it will return only the value of a randomly chosen attribute (`-e randomattrlist=cn:sn:uid:ou:l:fax:mobile`).

NOTE As with other operations, you may also perform randomized binding with every search operation.

Wildcard Searches

Because the `ldclt` threads perform synchronous LDAP operations by default, you should avoid searches using wildcards that return large numbers of entries. Depending on the size of your test directory, a search such as `-f cn=*` will block each thread for long periods of time, and each thread will receive search results containing the entire directory. This search will require large amounts of memory for each thread, possibly more than a total of 1 GB in certain cases.

NOTE The current version of the `ldclt` tool is optimized for doing exact searches that match a single entry.

Random Filters and RDNs

This section demonstrates exact searches, where the filter strings match only one entry. These filters are similar to RDNs (relative DNs) that determine a single entry under a given base DN. The `ldclt` tool has two interchangeable mechanisms, each of which can specify filter strings and RDNs:

- The `-f` *patternString* option uses other command-line options such as `-e incr,noloop -r0 -R99` to substitute values of `X` placeholders.
- The `-e -rdn=`*patternString* option uses inline definitions such as `INCRNNOLOOP(0;99;2)` to represent substitutions.

The `-f` *patternString* and `-e -rdn=`*patternString* have different syntax but you may use either one for any operations of the `ldclt` tool. You may use the `-rdn` option to define a filter string for searching or the `-f` option to define an RDN for another operation such as addition, as long as the result of each randomized substitution is valid for the respective operation. However, the two options are mutually exclusive on the command-line.

NOTE The `-f` *patternString* syntax is more limited and kept for backwards compatibility. The `-e rdn=`*patternString* format is more flexible and will be further expanded in the future.

-f patternString Parameter

The `-f` *patternString* parameter has the form *attribute=value* and the *value* may contain `X` characters as placeholders for random substitutions. Only the first, left-most sequence of `X` characters is replaced. (The capital letter `X` cannot be escaped for normal usage.) The following example uses the random filter string syntax to search for entries based on the mail attribute.

```
$ ldclt -h host -p port -b "ou=people,dc=example,dc=com" \
        -f mail=XXXXX@example.com -e esearch,random -r0 -R99999 \
        -e attrlist=cn:telephonenumber
```

The placeholders are replaced according to the options detailed in Table 11-6.

Table 11-6 -f patternString Parameter Options

Option and Parameter	Description Of Operation
`-e random -r`*low* `-R`*high*	Each `X` placeholder will be replaced with a digit of a randomly chosen number between *low* and *high*, inclusive. Numbers are padded to the left with zeros to fill all placeholders, and you should specify as many placeholders as digits in the largest number. If there are fewer placeholders than digits, the substitution will truncate the high digits.

Table 11-6 -f patternString Parameter Options *(Continued)*

Option and Parameter	Description Of Operation
`-e incr[,noloop][,commoncounter] -r`*low* `-R`*high*	Each X placeholder will be replaced with a digit of the current counter value. Numbers are padded to the left with zeros to fill all placeholders, and you should specify as many placeholders as digits in the largest number. High digits will be truncated if there are not enough placeholders. The counter is incremented after every operation, and it increases from *low* to *high*. By default each thread has a separate counter and will loop through all values indefinitely. Specify the `noloop` option to enumerate the counter values once and stop the thread afterwards. With the `commoncounter` option, all threads will increment the same counter and never substitute the same value.
`-e random,string[,ascii]`	Each X placeholder will be replaced with an arbitrary UTF-8 character to generate a random string. Use the `ascii` option to generate only 7-bit characters (hexadecimal values less than or equal to `0x7f`). Both UTF-8 and ASCII strings will be valid LDAP strings, including any escape characters that may be needed for special characters.

-e -rdn patternString Parameter

The `-e -rdn=`*patternString* parameter has the form *attribute*`:`*value* and the *value* may contain keyword placeholders for random substitutions. The following example uses the RDN keyword syntax to perform exactly the same type of searches as the filter string example used in "-e -rdn patternString Parameter" on page 154. Use single quotes (`''`) for the special syntax of the keywords in the shell.

```
$ ldclt -h host -p port -b "ou=people,dc=example,dc=com" \
        -e rdn='mail:[RNDN(0;99999;5)]@example.com' \
        -o object=foo.txt -e attrlist=cn:telephonenumber
```

The keywords determine the type of substitution and are replaced according to the rules detailed in Table 11-7.

Table 11-7 Substitution Keywords for -e -rdn patternString Parameter

Parameter	Description
`[RNDN(`*low*`;`*high*`;`*length*`)]`	This keyword will be replaced with a randomly chosen number between *low* and *high*, inclusive. Numbers are padded to the left with zeros or truncated on the left to have the exact *length* of digits.
`[RNDFROMFILE(`*filename*`)]`	This keyword will be replaced with a line chosen randomly from the file with the given *filename*. Each line of the file is substituted in whole, including any leading or trailing whitespace.

Table 11-7 Substitution Keywords for -e -rdn patternString Parameter *(Continued)*

Parameter	Description
[INCRN(*low*;*high*;*length*)] or [INCRNNOLOOP(*low*;*high*;*length*)]	This keyword will be replaced with the current value of a counter increasing from *low* to *high*, inclusive. Numbers are padded to the left with zeros or truncated on the left to have the exact *length* of digits. Each use of the keywords will represent a separate counter. In the first form, the counter will loop indefinitely, and in the second form, it will stop upon reaching the *high* value. Each thread will use a separate counter, or you may use the `-e commoncounter` option to use a single counter per keyword for all threads.
[INCRNFROMFILE(*filename*)] or [INCRNFROMFILENOLOOP(*filename*)]	This keyword will be replaced with a line chosen sequentially from the file with the given *filename*. Each line of the file is substituted in whole, including any leading or trailing whitespace. In the first form, the substitution will read through the file repeatedly, and in the second form, it will stop upon reaching the end of the file. Each thread will read the same file separately, or you may use the `-e commoncounter` option to read each line only once.
[RNDS(*length*)]	This keyword will be replaced with a string of the given *length* of randomly chosen ASCII characters. Random strings will be valid LDAP strings, including any escape characters that may be needed for special characters.

One limitation of the `-e rdn=`*patternString* option is that it *must* be used with a valid template file given by the `-o object=`*filename* option. This template file is needed by the randomization mechanism (“Random Entry Generation” on page 157) but has no effect if not required by the operation. The file `foo.txt` in this example contains the following minimal, valid template:

```
objectclass: person
cn: myCn
sn: mySn
```

Regardless of the format, search filters in performance tests usually include random numbers so that search operations are “scattered” over the directory. As we shall see in the following sections, additions, deletions and modify operations rely more often on counters and random strings to target specific entries.

Adding Entries

The entry creation functionality of the `ldclt` tool can be used to test the performance of the add operation and to populate a database for other tests. The command-line options provide many opportunities to randomize and customize the entries that are added to your directory. The following command is a simple example for entry addition:

```
$ ldclt -h host -p port -D "cn=Directory Manager" -w password \
        -f cn=MrXXXXX -b "ou=people,dc=example,dc=com" \
        -e add,person,incr,noloop,commoncounter -r0 -R99999
```

Performing additions requires write privileges in your directory, so you should specify the bind credentials of a user that can create the entries under the given base DN. This example will create 100,000 sequentially numbered entries. All threads use a common counter, so no two will try to create the same entry. This command will create simple `person` entries as reproduced in Code Example 11-2.

Code Example 11-2 LDIF of Created Entries

```
dn: cn=Mr00005,ou=people,dc=example,dc=com
objectclass: person
cn: Mr00005
sn: toto sn
```

The following sections demonstrate how to randomize the base DN and customize the generated entry.

Random Base DNs

When adding entries to a directory, you may want to place them under different suffixes to simulate real usage. To do this, the `ldclt` tool supports randomized base DNs in add operations; the same functionality is also available in modify and delete operations.

To use a different, numeric base DN with every operation, specify the `-e randombase` options and use `X` character placeholders in the `-b` *baseDN* parameter. Only the first, left-most sequence of `X` characters is replaced. (The capital letter `X` cannot be escaped for normal usage.) For example, each thread of the following command will create `person` entries in any one of the 1,000 possible department base DNs:

```
$ ldclt -h host -p port -D "cn=Directory Manager" -w password \
        -b "ou=DeptXXX,dc=example,dc=com" \
        -e randombase,randombaselow=0,randombasehigh=999 \
        -f cn=MrXXXXX -e add,person,incr,noloop -r0 -R99999
```

The values substituted into the base DN are generated independently of other random substitutions into bind DNs, filters, or RDNs. See “Random Filters and RDNs” on page 152 for more information.

Create Missing Nodes

When creating entries with random base DNs, the tool has the ability to automatically create the corresponding nodes if they do not already exist. This allows you to populate large directories of any depth with the add operation. The randomly named nodes are also created during rename operations. See “modRDN Operations” on page 162 for more information.

In order to create node entries automatically, the naming attributes of the base DN must be cn, o or ou. Depending on the naming attribute, the `ldclt` tool will create intermediate nodes of the following object classes:

```
cn  -> OrganizationalRole
o   -> organization
ou  -> organizationalUnit
```

For example, the add operation may try to create the entry with the randomized DN `cn=Mr`*`00007`*`,ou=Dept`*`123`*`,dc=example,dc=com`. Assuming the base DN does not exist in the directory yet, the operation will first create the following node:

```
dn: ou=Dept123,dc=example,dc=com
objectclass: organizationalUnit
ou: Dept123
```

CAUTION Due to the implementation of the missing node mechanism, the entry that triggers the creation of a missing node is *not* created. In the above example, the DN `cn=Mr`*`00007`*`,ou=Dept`*`123`*`,dc=example,dc=com` will not be added. Only subsequent entries in this node will be added, now that the node exists.

Random Entry Generation

Just as there are two ways to specify filters and RDN strings, there are two different mechanisms to define the entries that the tool will add to the directory. The simpler method seen in previous examples creates simple entries from fixed templates. The more powerful method allows you to specify the object class template and perform many random substitutions.

To generate simple person entries, use the following command:

```
$ ldclt -h host -p port -D "bindDN" -w password \
        -f cn=patternString -b "baseDN" \
        -e add,person [randomizingOptions]
```

The created entries will have the contents detailed in Code Example 11-3, with any randomizations substituted into the *patternString* and *baseDN*.

Code Example 11-3 Contents of Simple Entry in LDIF

```
dn: cn=patternString,baseDN
objectclass: person
cn: patternString
sn: toto sn
```

To generate simple inetorgperson entries, use the following command:

```
$ ldclt -h host -p port -D "bindDN" -w password \
        -f cn=patternString -b "baseDN" \
        -e add,inetorgperson,imagesdir=Path [randomizingOptions]
```

The created entries will have the contents detailed in Code Example 11-4, with any randomizations substituted into the *patternString* and *baseDN*.

Code Example 11-4 Contents of inetorgperson Entry in LDIF

```
dn: cn=patternString,baseDN
objectclass: inetorgperson
cn: patternString
sn: toto sn
jpegPhoto: randomBinaryFileFromPath
```

Specifying the emailperson option will generate entries that are identical to the inetorgperson entries, except for the value of the objectclass attribute. In all cases, the *patternString* must use either sn (surname) or cn (common name) as the naming attribute. If sn is used, cn will contain the value toto cn.

More complex entries can be created from custom templates with the `-e rdn=`*patternString*`,object=`*filename* options. The template file provides all of the attributes for the entry with the following syntax:

```
# comments begin with a '#'
#
Attribute-> AttrName: Value+
AttrName-> string
Value    -> string | Variant
Variant  -> [Variable=Keyword] | [Variable] | [Keyword]
Variable -> A | B | C | D | E | F | G | H
Keyword-> RNDN(low;high;length) |
                RNDFROMFILE(filename) |
                INCRN(low;high;length) |
                INCRNNOLOOP(low;high;length) |
                INCRFROMFILE(filename) |
                INCRFROMFILENOLOOP(filename) |
                RNDS(length)
```

The keywords are the same as defined in "Random Filters and RDNs" on page 152. Variables definitions should be used in the RDN *patternString* so that the same values can be used as the naming attributes inside the template file. For example, the following command and template file will generate 100,000 sequentially numbered entries:

```
$ ldclt -h host -p port -D "bindDN" -w password \
        -b "ou=people,dc=example,dc=com" -I 68 \
        -e object=inet.txt,rdn='uid:[A=INCRNNOLOOP(0;99999;5)]' \
        -e add,commoncounter
```

The associated template file, `inet.txt` (Code Example 11-5), uses the variable `A` and other keywords to create a realistic entry:

Code Example 11-5 inet.txt File

```
# inet.txt - template for a more complete inetOrgPerson
#
objectclass: inetOrgPerson
sn: [B=RNDFROMFILE(installDir/data/dbgen-FamilyNames)]
cn: [C=RNDFROMFILE(installDir/data/dbgen-GivenNames)] [B]
password: test[A]
description: user id [A]
mail: [C].[B]@example.com
telephonenumber: (555) [RNDN(0;999;3)]-[RNDN(0;9999;4)]
jpegPhoto:: /9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAoHBwkHBgoJCAkLCwoM
 DxkQDw4ODx4WFxIZJCAmJSMgIyIoLTkwKCo2KyIjMkQyNjs9QEBAJjBGS0U+Sjk
 /QD3/2wBDAQsLCw8NDx0QEB09KSMpPT09PT09PT09PT09PT09PT09PT09PT
```

Code Example 11-5 inet.txt File *(Continued)*

```
09PT09PT09PT09PT09PT09PT09PT3/wAARCAAiACIDASIAAhEBAxEB/8QAG
gAAAgMBAQAAAAAAAAAAAAAAAYDBAUBB//EACoQAAIBAwMDAgYDAAAAAAAAAEC
AwQFEQASIQYTMRVBFBYiI1FhBxdx/8QAGQEAAwADAAAAAAAAAAAAAAAAAAQGAQM
F/8QAJhEAAQMCBAYDAAAAAAAAAAAAAQACEQMEBRIhQRMxYXGh8BQikf/aAAwDAQ
ACEQMRAD8A9Xul0W3okccffrJsiCANjdjGWY87UGRlsHGQACxVSuyx1MkNO3Ud8
eKSdlQU9JOaSHvEcLGykSsfPDOQTk7RwFloEjr71cru6K0hmaigdlG9IYm2Mn+G
VZW/JBXPgAKv8o9IXTqv0j0tIm7EkiS7327A+36v2BtOcZPIwDqRvsTfXvPitqc
NgkE7yAd++iYayG5olNi2UwHdQ3W7U0h4LmseoyPxtn3qPbkDPHnBOb9ou9Z3lo
b4tOlW7N2JqcMIqhRk4AYkrIFGShJyMlScNtRPm+6f3H8v74vTe3s7fb5z2e7u3
ec54/GPbPOm6+W03O2SRxLF8ZF96jkkGRDOvMb+D4bHscjIIIJBUtsSu7CrTFw/
M14B1M6Hqd1ksa4GE0aNVbZXxXW1UlwgV1iqoUnQOAGCsoYZxnnB0auUslmghmF
LerdBP2KuGtqQJwgcI0pM6MAfOFmTIPuCPHJnpbzDJUpR1g+ErzkCCTOJCBkmJi
AJBjnK8gEbgp41eu1ok9RW80CvJVpD2JafuELURBiwAydokUlipPB3MpIzuXNgu
Nq6hpzRTpG0rxiSa3VkYE0YyD9yJuRgkc4x4IJBB1BYxYPoV31C0ljjMjYnnPvY
jVNU3SIXbrcbNZJluFyemhqXjMSSbN0zoCCVUAFmAJBIGceTqChvYbpyrv08hkp
Cj1UcMYUtDEqDKE8ZfKsSD4ZiuSFyV7rPoyzJRLVUj2y0S5EZErLTwTeSASB9Lj
kggZ4IPsV0Oh+k5Xt0LV8dVBTx1nxhjkDRNPOmFU7eGES7FYBwGdgGIQABiywyn
eU28MkmdZHIe9fzVb38JtIOzfaeUeZ98SXSwUEtq6ctlvnZGlpaSKByhJUsqBTj
OOMjRrQ0avVzkaq19sobrAsNxo6ariVt4jqIlkUNgjOCDzgnn96NGhChoLBaLVO
01utVDSSsuwyU9OkbFcg4yAOMgcfrWho0aEI0aNGhC//Z
```

The entry creation mechanism automatically adds the naming attribute in the RDN to the new entry. In this example, the attribute and substituted value `uid: [A]` will be added to every new entry.

CAUTION Do not specify naming attributes in the template file, otherwise they will be duplicated in the new entry and cause an LDAP error.

Generating LDIF Files

The add operation can be used to populate large databases for testing. The same random entry generation mechanisms can be used to create large database files in LDIF (LDAP Data Interchange Format). Instead of the `-e add` option, use the `-e genldif=`*filename*`[,append]` options to specify the output file and whether it should be appended to. No host, port, or bind credentials are needed because no LDAP operations are performed on a directory. The following command creates a file of 100,000 sequentially numbered users with the template file in Code Example 11-5.

```
$ ldclt -b "ou=people,dc=example,dc=com" \
        -e object=inet.txt,rdn='uid:[A=INCRNNOLOOP(0;99999;5)]' \
        -e genldif=100Kinet.ldif,commoncounter
```

TIP For creating entries with random DNs, such as names, it is common practice to have a non-looping counter in the template file to limit the contents of the LDIF file. By placing this counter in the `description` attribute, for example, its high and low range will determine the number of entries created by each thread.

Only the target entries are generated in LDIF, missing nodes are not created. Use the command twice with the append option and different templates to generate LDIF files with numerous intermediate nodes. To create an entire tree in LDIF, edit an empty file manually to add a root entry at the beginning, and then append as many levels of nodes or leaves to the file as you wish.

Modify Operations

The `ldclt` tool performs simple LDAP modify operations that are limited to changing the value of a single attribute. The `-e attreplace` option specifies the attribute name which will be modified and a new value that may be randomized for every operation.

NOTE Performing modifications requires write privileges in your directory, so you should specify bind credentials, whether randomized or not, that can modify the entries under the given base DN.

The following command uses the Directory Manager credentials to modify the `description` attribute of randomly selected entries. The `X` placeholders in the new value string will be replaced with random characters, independently of any other *patternString*, bind DN or base DN substitutions:

```
$ ldclt -h host -p port -D "cn=Directory Manager" -w password \
        -b "ou=people,dc=example,dc=com" \
        -e rdn='uid:[RNDN(0;99999;5)]' -I 32 \
        -e attreplace='description: random modify XXXXX'
```

Use the `-I 32` option to ignore error 32, `LDAP_NO_SUCH_OBJECT`, that will occur if the randomization defines a DN for an entry that does not exist.

The following example performs modification under several nodes of the directory tree, using options explained in “Random Base DNs” on page 156. In this case multiple threads may access the same entry if they randomly happen to be in the same node, but concurrent modifications should not cause an error. Again, the new attribute value will contain a random string generated for each operation.

```
$ ldclt -h host -p port -D "cn=Directory Manager" -w password \
    -b "ou=DeptXXX,dc=example,dc=com" \
    -e randombase,randombaselow=0,randombasehigh=999 \
    -e rdn='cn:Mr[INCRNNOLOOP(0;99999;5)]' -I 32 \
    -e attreplace='sn: XXXXXX'
```

Delete Operations

The command for performing delete operations is similar to that for performing modifications. The bind credentials must have write privileges to remove entries, and you may use the randomization features to determine the base DN and RDN of the target entry of each operation. The following command deletes random entries from a directory. Use the `-I 32` option to ignore error 32, `LDAP_NO_SUCH_OBJECT`, that is likely to occur because entries may be randomly selected to be deleted a second time.

```
$ ldclt -h host -p port -D "cn=Directory Manager" -w password \
        -b "ou=DeptXXX,dc=example,dc=com" \
        -e randombase,randombaselow=0,randombasehigh=999 \
        -e delete,rdn='cn:Mr[RNDN(0;99999;5)]' -I 32
```

The delete operation supports the more flexible `-e rdn=`*patternString* for specifying the RDN of the target entry. The following command will delete all entries in branch of your directory, providing you specify the entire range of sequential numbers:

```
$ ldclt -h host -p port -D "cn=Directory Manager" -w password \
        -b "ou=people,dc=example,dc=com" -e delete -I 32 \
         -e rdn='uid:[INCRNNOLOOP(0;99999;5)]',commoncounter
```

modRDN Operations

The rename functionality is similar to that for modifications: the bind credentials require write privileges whether randomized or not, and you can randomize both the base DN and the RDN.

The `ldclt` tool performs LDAP `modRDN` operations on target entries by generating a second randomized RDN as the new name. Because the same mechanism is used to generate the new RDN, sequential numbering of target entries is not recommended. The following command uses the `-e rdn=`*patternString* options to specify a random RDN for the target entry.

```
$ ldclt -h host -p port -D "cn=Directory Manager" -w password \
        -b "ou=people,dc=siroe,dc=com" -I 32 -I 68 \
        -e rename,rdn='uid:[RNDN(0;99999;5)]'
```

If the base DN is also randomized, you may specify the `-e withnewparent` option to also generate a new random parent. The following command shows an example using this option:

```
$ ldclt -h host -p port -D "cn=Directory Manager" -w password \
        -b "ou=DeptXXX,dc=example,dc=com" -I 32 -I 68 \
        -e randombase,randombaselow=0,randombasehigh=999 \
        -e rename,withnewparent,rdn='cn:Mr[RNDN(0;99999;5)]'
```

Since the new parent DN is randomly generated, it may not exist in your directory. However, the `ldclt` tool will create the parent node automatically when needed. (See "Create Missing Nodes" on page 157 for more information.) Due to the implementation of the missing node mechanism, the entry that triggered the creation of a missing node is *not* renamed, but subsequent reparenting operations will use the new node. Also, both examples use the `-I` option to ignore errors that may occur when renaming random RDNs.

Troubleshooting

This section explains the most common messages and errors encountered when using the `ldclt` tool and gives potential solutions.

No Thread Activity

When a thread has not performed any successful operations during a certain time, the ldclt tool displays the following message:

```
T002: No activity for 40 seconds --     3 in row, total     17
```

This sample message contains the following information:

- Thread 2 has not performed a successful operation in the past 40 seconds.
- This is the third such consecutive message for this thread, for a total of 120 seconds of inactivity.
- Since the beginning of the current process, this thread has had 17 x 40 seconds of inactivity.

The most common reason for this message is that the server is overloaded and the thread is waiting for an operation to finish. When the server is overloaded, all threads will have short moments of inactivity, perhaps many in total, but never very many in a row. To account for server load, you may use the `-i` *intervals* option to allow longer periods of inactivity.

Another common cause for this message is that very few operations are succeeding. For example, when deleting randomly generated RDNs, successful delete operations will become rarer unless another process is creating entries with the same RDN pattern. Use the `-e counteach` option to account for unsuccessful operations and avoid the message about false inactivity.

A more unlikely scenario occurs if only one thread constantly reports inactivity messages, as shown in the following output sample:

```
T002: No activity for 40 seconds --     1 in row, total     1
T002: No activity for 40 seconds --     2 in row, total     2
T002: No activity for 40 seconds --     3 in row, total     3
...
T002: No activity for 40 seconds --     8 in row, total     8
...
```

In this case, it is likely to be waiting for a response which may be blocked for an abnormal condition such as a library or server bug. Since the condition may override any time-out mechanism on the server side, the thread will remain inactive until the `ldclt` process is stopped.

Exit Status 3

The `ldclt` tool maintains counters for most common errors and will terminate with exit status 3 at the end of the interval where the error limit is exceeded for any one error. The following output shows this situation:

```
Max error limit reached - exiting.
Global average rate:    0.00/thr  (   NaN/sec), total:      0
Global number times "no activity" reports: never
Global error 32 (No such object) occurs  1020 times
Ending at Mon Apr  9 17:32:51 2001
Exit status 3 - Max errors reached.
```

By default the error limit is 1000 for each type of error. To avoid exiting in this condition, specify one of the following options:

- `-I` *errorCode* will ignore any error with the given code, so that it will never be counted. Use this option with operations on random entries that may return errors, such as deleting a non-existant RDN.

- `-E` *occurences* will set the error limit to the given number of occurences. This option can be used to raise or lower the error limit. For example, `-E 0` will make the tool exit for any error that is not ignored.

See “Error Handling” on page 146 for more information.

Error 32

Error code 32 (`LDAP_NO_SUCH_OBJECT`) occurs commonly for searches, modifications, deletions, and rename operations. In these cases, the error is expected when using randomized base DN or RDN values that may not be present.

This error may also occur during add operations due to a DN syntax error. The missing node functionality will recursively attempt to create the parent node and fail with error 32. The following output shows this situation:

```
T007: Cannot add (cn=,00007,ou=people,dc=example,dc=com),
error=32 (No such object)

T007: Don’t know how to create entry when rdn is "00007,ou=people"

T007: Cannot create the intermediate nodes for
cn=,00007,ou=people,dc=example,dc=com
```

To this error, check the RDN for illegal characters in the *patternString* or possibly in the file containing lists of DNs to add with the `RNDFROMFILE` keyword.

Error 65

Error code 65 (`LDAP_OBJECT_CLASS_VIOLATION`) occurs when trying to perform an addition with an invalid template in the `-e object` option. For example:

```
T004: Cannot add (uid=00007,dc=example,dc=com), error=65 (Object
class violation)
```

The server returns this error because the entry created from the template is not allowed by the directory’s schema. The exact schema violation will be given in the server’s error logs, for example:

```
[14/Feb/2002:19:38:52 -0700] - Entry "uid=00007,dc=example,dc=com"
-- attribute "mailAlternateAddress" not allowed
```

To solve this problem, you may do one of the following:

- Modify the template file so that `ldclt` will generate valid entries.

- Modify the directory schema to accept new objects or new attributes.
- Turn off schema checking in your directory.

With Directory Server versions 4.*x*, extra space characters in the template file are not ignored. This will cause error 65 that can be solved by removing extra spaces from the template file. The following example shows an extra space before the closing parenthesis in the error log:

```
[04/Apr/2001:15:40:28 +0200] - Entry (cn=Mr00007,ou=people,
dc=example,dc=com) unknown object class (person )
```

Error 81

Error code 81 (`LDAP_SERVER_DOWN`) can be misleading because it may occur when a thread cannot establish a connection. The error message is the following:

```
T030: Cannot ldap_search(), error=81 (Can't contact LDAP server)
```

This actually means that the client cannot connect to the server for some communication reason, such as no more operating system-level connections are available on either the client or the server. It does not necessarily mean the directory server is not running.

Possible causes include:

- If the error occurs from time to time on an arbitrary thread, the most likely reason is that either your client or server machine has a memory limitation, such as no more swap space.
- If the error occurs in LDAP bind operations when performing many bind operations, you should tune the TCP/IP layers of both your client and server machines. See Chapter 10, "The idsktune Optimization Tool" for more information.
- This error is more likely to occur on slow machines, especially when using the `-e bindeach` option to establish a connection for every operation.

You may use the `-I 81` option to ignore this error if it occurs during a bind. Instead of exiting, the corresponding thread will attempt to rebind, after a 1 second idle period to avoid overloading the server. (See the `-e dontsleeponserverdown` option in Table 11-1 on page 138.) Error 81 is fatal to the calling thread if it occurs on any other operation, or if it is not ignored during the bind.

Error 89

Error code 89 (`LDAP_PARAM_ERROR`) can occur when calling the functions of the LDAP SDK for C with bad parameters. For example:

```
T030: Cannot ldap_parse_result(), error=89 (Bad parameter to an
ldap routine)
```

This error usually occurs in the `ldclt` tool when trying to parse the result of an operation that failed due to a connection error. Because the operation failed, there is no result to parse and the error is compounded. This error often occurs after error 81 described previously and can also be ignored by using the `-I 89` option.

Error 91

Error code 91 (`LDAP_CONNECT_ERROR`) is returned when there is no process accepting connections on the given port and the given machine. It usually means there is no directory server responding at the given `-h` *host* and `-p` *port* options. For example:

```
T003: Cannot ldap_simple_bind_s (cn=directory manager,password),
error=91 (Can't connect to the LDAP server)
```

You may use the `-I 91` option to ignore this error if it occurs during a bind. Instead of exiting, the corresponding thread will attempt to rebind, after a 1 second idle period to avoid overloading the server (see the `-e dontsleeponserverdown` option in Table 11-1 on page 138). Error 91 is fatal to the calling thread if it occurs on any other operation, or if it is not ignored during the bind.

Undocumented Options

The `ldclt` tool has several `-e scalab01` options that are used internally for specialized test scenarios. These options implement behavior for a custom simulation that remains undocumented. These options are unsupported and will cause unexpected behavior.

Chapter 12

The rsearch Search Tool

The `rsearch` tool is a multi-threaded program that measures the performance of LDAP search, compare, modify, delete, and authentication operations. It performs operations continuously and computes an average operation rate at regular intervals. This chapter provides instructions on how to use the `ldclt` tool. It contains the following sections:

- Overview
- Command Usage
- Sample Output
- Command-Line Examples

Overview

The `rsearch` tool (repeated search) is an LDAP client that performs repeated LDAP operations. The DSRK and its updates include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory. It is written in C using the Sun™ ONE LDAP SDK for C and may be used to measure performance on any LDAP directory.

NOTE As with all measures of performance, results depend upon many factors, such as the options and parameter values given, directory configuration, machine load, and network traffic, and should be analyzed accordingly. In order to obtain accurate performance measurements, both client and server machines should be dedicated to the performance test and properly tuned. (See Chapter 10, “The idsktune Optimization Tool” for tuning information.)

Command Usage

The `rsearch` command launches a number of threads that perform synchronous operations on the given directory server.

NOTE The threads are simple loops that perform the same operation over and over as quickly as possible.

At regular intervals (every 10 seconds by default) the tool displays the statistics collected for operations completed during the elapsed interval. The default operation is to perform a search from a filter defined on the command-line. (In order to simulate more realistic directory usage, you may provide a filter file with multiple filter strings that the tool will select from randomly.) The `rsearch` tool will also measure the performance of compare, modify, delete, and authentication operations. These require you to provide a file containing distinguished names (DNs) or UIDs for entries that will be randomly selected to perform each operation. The output for these operations is:

- The total number of operations
- The average number of operations per thread
- The number of operations per second
- The inverse in milliseconds per operation

Syntax

The syntax of the `rsearch` tool on the command-line takes the following form:

```
rsearch -D "bindDN" -w password -s "suffix" -f "filter" [ options ]
```

Where:

- *bindDN* and *password* are the bind credentials with sufficient permissions to search or modify the directory.
- *suffix* is the base DN for the search or modify operations.
- *filter* is an RFC 2254-compliant LDAP search filter. (See “LDAP Search Filters” in Chapter 4 of the *Sun ONE Directory Server Getting Started Guide*). The filter may use the `%s` syntax to include strings from a file given with the `-i` option.
- *options* are the command-line options and parameters described in “Options” on page 171.

NOTE Command-line parameters such as DNs and filters should be enclosed in double quotes ("...") if they contain special characters for the shell. When the required options (`-D`, `-w`, `-s`, and `-f`) are not needed for a given operation, they may be given as an empty string ("").

Options

The `rsearch` options and parameters are described in Table 12-1. The `rsearch -\?` command when run on the command-line will display text that briefly describes all of the command-line options.

Table 12-1 Command-Line Options for `rsearch`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. The default is `localhost`.
`-p`	*port*	Specify the port number when accessing the directory server host. The default is 389.
`-D`	*bindDN*	Specify a bind DN for all operations, usually in double quotes ("") for the shell. Depending on its access permissions, the given bind DN may affect authentication and search performance.
`-w`	*password*	Specify the password for the bind DN.
`-s`	*suffix*	Specify the suffix to use as the base DN for all operations, usually in double quotes ("") for the shell.
`-f`	*filter*	Specify a filter for search operations, usually in double quotes ("") for the shell. This option can be used with an input file, for example, the filter `cn=%s` will use a random string from a file specified by the `-i` option.
`-i`	*filterFile*	Give the name of the file containing filter strings. The filter will be selected randomly from this file. See “Filter File Format” on page 173 for more information.
`-S`	*scope*	Specify the scope of a search. The *scope* parameter may have one of the following values: • `0` - For searching only the base entry. • `1` - For searching one level below the base entry. • `2` - For searching all levels below the entry. This is the default.
`-A`	*attributes*	A comma separated list of attribute names to specify which attribute values will be returned by searches.

Table 12-1 Command-Line Options for `rsearch` *(Continued)*

Option	Parameter	Purpose
`-N`		No operation: using this flag, the tool will only bind to the server, instead of binding and performing a search, which is the default. The `-c`, `-d`, and `-m` options will be ignored when this flag is used.
`-b`		Perform a bind before every operation (and unbind afterwards).
`-u`		Do not unbind from the server, just close the connection.
`-L`		Set the linger mode on the TCP socket to avoid too many leftover closed connections.
`-y`		Set the `TCP_NODELAY` mode on the TCP socket.
`-t`	*threads*	Specify the number of threads that `rsearch` will use. Use the `-v` option for verbose output including measurements from each thread. The default is a single thread.
`-j`	*seconds*	Specify the sampling interval, in seconds; the default is 10. `rsearch` repeats the given operation as many times as possible during the interval and prints results after each interval elapses.
`-T`	*seconds*	Specify the time limit, in seconds, after which `rsearch` will display the cumulated average and stop. When this option is not specified, `rsearch` will run until its process is terminated.
`-V`		Alternate output at every interval: display only the rate and cumulated average rate of operations per thread.
`-v`		Verbose output at every interval: gives the measurements from each thread, including the minimum and maximum operation times observed, as well as the average for all threads.
`-q`		Quiet output mode: the measurements for each interval will not be displayed. If `-T` is specified, only the final average will be shown.
`-B`	*UIDfile* or *DNfile*	Specify the file needed for input to the compare, delete, modify, and authentication operations. Must be used in conjunction with the `-c`, `-d`, `-m` or `-x` flags, respectively, to specify the operation. See "DN and UID File Formats" on page 174 for more information.
`-c`		Perform compare operations on the `uid` attribute of entries chosen randomly from a DN file specified by the `-B` option. Comparisons are set up to be randomly true or false and verified accordingly.
`-d`		Perform delete operations on entries chosen randomly from a DN file specified by the `-B` option. Note that `rsearch` does not replace these entries, meaning that delete operations will cause an error when a DN is randomly selected the second time. This will not affect `rsearch`, but the statistics will be skewed unless the entries are otherwise restored or replaced.

Table 12-1 Command-Line Options for `rsearch` *(Continued)*

Option	Parameter	Purpose
`-m`		Perform modify operations on the `description` attribute, assuming it is not indexed. The entries to be modified are chosen randomly from a DN file specified by the `-B` option.
`-M`		Perform modify operations on the `telephonenumber` attribute, assuming that it is indexed. The entries to be modified are chosen randomly from a DN file specified by the `-B` option.
`-x`		Perform bind operations using a DN and its UID as a password for authentication. The DN is selected randomly from a UID or DN file specified by the `-B` option. See "DN and UID File Formats" on page 174.
`-\?`		Display the usage help text that briefly describes all options.

Filter File Format

The `-i` option is followed by the name of a text file containing filter strings. Each line of the file is taken to be a filter string and may contain spaces, as follows:

```
Filter string 1
Filter string 2
...
```

When using a filter file, you must use the `-f` option with a filter containing the `%s` placeholder. When performing each search, the placeholder will be textually replaced by a filter string randomly selected from the filter file. The filter may contain only one placeholder, for example, `-f "cn=%s"`. Alternatively, you may specify `-f "%s"` on the command-line and give complete filters in the filter file. This will allow you to perform more complex searches.

TIP When using a filter file, it should contain the appropriate strings for searching in your directory. For example, you may want to measure exact string matching searches, searches that fail, or compound search expressions. Also the number of filter strings will determine how often the same search is repeated: few strings imply that search results are more likely to be retrieved from the directory server's cache.

DN and UID File Formats

The `-B` option specifies a file containing either DN and UID values or UID values alone, as shown in Table 12-2. DN files are used as input for the compare, modify, and delete operations. Both DN and UID files may be used as input for binding when performing authentication and search operations. See "Measuring Bind and Authentication Operations" on page 175, and "Random DN Bind and Search Rate" on page 178 for more information.

Table 12-2 DN or UID File Formats

DN File Format	UID File Format

`DN: dn_string`
`UID: uid_string`
`DN: dn_string2`
`UID: uid_string2`
`...`
	`UID: uid_string`
	`UID: uid_string2`
	`...`

In these formats, the `DN:` and `UID:` keywords must be followed by at least one space. Everything after that space is taken literally; do not use quotes unless you want them to appear in the string.

CAUTION When used for binding by specifying the `-x` option, the two file formats are treated differently. If you use the DN file format, authentication for the bind will use the DN as the bind DN and the UID as the password. If you have a file containing only UIDs, `rsearch` will use its `-D` *bindDN* and `-w` *password* parameters to bind and search for an entry with the given UID. It will find the DN of that entry, and then bind again with this DN, using the UID as the password.

Sample Output

Code Example 12-1 is sample output that might be retreived from the `rsearch` tool. The given `Rate` is the average number of operations per thread over the elapsed interval (10 seconds by default or use the `-j` *seconds* option to specify a different interval). All data on an output line concerns only the elapsed interval. Use the `-V` option to display a running average of operations per thread per interval.

Code Example 12-1 Sample Output From rsearch Tool

```
Rate: 648.00/thr (129.60/sec = 7.7160msec/op ), total: 1296 (2 thr)
Rate: 645.00/thr (129.00/sec = 7.7519msec/op ), total: 1290 (2 thr)
Rate: 642.50/thr (128.50/sec = 7.7821msec/op ), total: 1285 (2 thr)
...
Final Average rate: 130.74/sec = 7.6488msec/op, total:  1285
```

If you specify a time limit with the `-T` *seconds* option, the final average rate is computed and displayed; otherwise, the command runs until you terminate it. This average is totaled over all threads, giving the absolute operation rate and its inverse, the average time to complete each operation.

Command-Line Examples

The following sections give examples of `rsearch` command usage for measuring the performance of various scenarios. You will need to adapt these examples to your environment based on the following:

- The *hostname* and *port* placeholders should be replaced with the hostname and port number of your directory. When significant, the *suffix* should represent the contents of the directory tree or subtree you wish to test.
- For meaningful results, thread numbers and time limit options should be scaled according to your directory's expected load.
- You will need to provide DN and UID files that correspond to the contents of your directory. (See "DN and UID File Formats" on page 174 for more information.)

Measuring Bind and Authentication Operations

The following examples will measure bind and authentication performance in your LDAP directory. These commands perform only bind and unbind operations. They do not perform any searches, except when part of the binding procedure.

NOTE In all of the bind and authentication examples, the `-s` *suffix* and `-f` *filter* options have no effect because no search is performed. They are given as empty strings (`""`) because they are required on the command-line.

Anonymous Bind Rate

This command will bind anonymously (`-D "" -w ""`), repeat binding with no other operations (`-N -b`), avoid opening too many connections (`-L`), use a single thread (no `-t` option), display statistics every 10 seconds (no `-j` option), and finish in about 100 seconds (`-T 100`).

```
$ rsearch -h hostname -p port -s "" -f "" \
          -D "" -w "" -N -b -L -T 100
```

Random DN Authentication Rate

This command will bind repeatedly, each time as a random DN found in the DN file (`-B DNfile -x`). It uses the DN's UID from that file as the password, repeats binding with no other operations (`-N -b -L`), and runs indefinitely (no `-T` option).

```
$ rsearch -h hostname -p port -s "" -f "" -D "" -w "" \
          -B DNfile -x -N -b -L
```

NOTE The `-D ""` and `-w ""` options are required on the command-line but have no effect.

Random UID Authentication Rate

This command is a typical authentication scenario where the client must first find the DN corresponding to a UID before binding. In each bind sequence, this command will select a random UID from the given UID file (`-B UIDfile -x`), bind anonymously (`-D "" -w ""`) to find the DN of the corresponding entry, and then bind as this DN using the UID for authentication. It will repeat the binding sequence without performing searches until the process is killed (`-N -b -L`).

```
$ rsearch -h hostname -p port -s "" -f "" \
          -D "" -w "" -B UIDfile -x -N -b -L
```

NOTE You may need to set the `-D` and `-w` options for valid authentication if your directory does not allow anonymous binding.

Root DN Bind Rate

This command will bind as root DN (`-D "cn=Directory Manager" -w password`), and repeat binding with no other operations (`-N -b -L`) for about 100 seconds (`-T 100`).

```
$ rsearch -h hostname -p port -s "" -f "" \
          -D "cn=Directory Manager" -w password -N -b -L -T 100
```

Measuring Search Operations

The following examples will measure search performance in your directory server. All of the searches are performed within a single bind, so the results are those of the search operation alone, assuming factors such as machine load and network traffic remain constant.

In these examples, you will need to provide the suffix and filter strings for valid entries in your test directory. (See “Filter File Format” on page 173 for more information.) Filter files may also contain the following different kinds of filter strings that may give different performance results:

- Strings containing wildcards will give measurements that reflect substring search rate.
- Strings without wildcards will give measurements that reflect the search rate for exact matches.
- Filter strings may also include operators other than equality, such as ranges that are greater than (>=) or less than (<=) a given value, or approximate spelling searches (~=).

Simple Search

This command will bind once as the given DN (`-D` *bindDN* `-w` *password*) and search repeatedly for entries under the `"dc=example,dc=com"` suffix that contain the `"cn=*jones*"` substring. Because every search uses the same filter and returns the same entries, entry caching in the directory server will influence results. In all of the following examples, the use of a filter file helps provide more realistic measurements.

```
$ rsearch -h hostname -p port -s "dc=example,dc=com" \
          -D "bindDN" -w password -f "cn=*jones*"
```

Search Rate Using Anonymous Bind

This command will perform a single anonymous bind (`-D "" -w ""`) and search repeatedly for random surnames taken from a filter file (`-f "sn=%s" -i` *filterFile*).

```
$ rsearch -h hostname -p port -s "dc=example,dc=com" \
          -D "" -w "" -f "sn=%s" -i filterFile
```

Search Rate Using DN Bind

This command will bind once as the given DN (`-D` *bindDN* `-w` *password*) and search repeatedly for entries matching random surnames taken from a filter file (`-f "sn=%s" -i` *filterFile*).

```
$ rsearch -h hostname -p port -s "dc=example,dc=com" \
          -D "bindDN" -w password -f "sn=%s" -i filterFile
```

Specific Attribute Search Rate

This command will retrieve only the `givenName` and `mail` attributes of entries matching random surnames taken from a filter file (`-f "sn=%s" -i` *filterFile*).

```
$ rsearch -h hostname -p port -s "dc=example,dc=com" \
          -D "" -w "" -f "sn=%s" -i filterFile -A givenName,mail
```

Measuring Bind and Search Operations

The following commands are similar to the search examples, except that `rsearch` will bind before every search operation and unbind afterward. These examples mimic the usual behavior of directory clients and thus measure the average time taken to serve a directory client. Again, if filter files contain strings with wildcards, performance will reflect substring search rates, otherwise it will reflect exact search rates.

TIP To further resemble actual search operations, multiple threads can simulate load on the server. Set the thread option to model the expected load on your directory.

Anonymous Bind and Search Rate

This command will create 10 threads (`-t 10`), each of which will repeatedly bind, search, and unbind (`-b`), while avoiding too many open connections (`-L`). It always uses anonymous binding (`-D "" -w ""`) and searches for random surnames (`-f "sn=%s" -i` *filterFile*) in entries under the `"dc=example,dc=com"` suffix.

```
$ rsearch -h hostname -p port -s "dc=example,dc=com" \
          -D "" -w "" -f "sn=%s" -i filterFile -b -L -t 10
```

Random DN Bind and Search Rate

This command will create 10 threads (`-t 10`), each of which will repeatedly bind, search for entries under the `"dc=example,dc=com"` suffix, and unbind (`-b`), while avoiding too many open connections (`-L`). It always binds with a DN randomly selected from the DN file (`-B` *DNfile* `-x`) and uses the DN's UID from that file as the password. (See "DN and UID File Formats" on page 174.) The `-D ""` and `-w ""` options are required on the command-line but have no effect.

```
$ rsearch -h hostname -p port -s "dc=example,dc=com" -D "" -w "" \
          -B DNfile -x -f "sn=%s" -i filterFile -b -L -t 10
```

Random UID Bind and Search Rate

This command is identical to the previous one for measuring random DN bind and search rate, except it uses a UID file. (See "DN and UID File Formats" on page 174.) Here, in each repeated search sequence, the command will first select a random UID from the UID file (`-B` *UIDfile* `-x`), bind anonymously (`-D "" -w ""`) to find the DN of the corresponding entry, and then bind as this DN using the UID for authentication before performing the search.

```
$ rsearch -h hostname -p port -s "dc=example,dc=com" -D "" -w "" \
          -B UIDfile -x -f "sn=%s" -i filterFile -b -L -t 10
```

NOTE You may need to set the `-D` and `-w` options for valid authentication if your directory does not allow anonymous binding.

Measuring Compare Operations

The following examples will measure the performance of compare operations. The `-s ""` and `-f ""` options have no effect but are required on the command-line.

Compare Rate

This command will create two threads (`-t 2`), each of which will bind once using the given *bindDN* and *password* (use `-D "" -w ""` for anonymous binding), repeatedly perform a compare operation (`-c`) on the `uid` attribute of random DNs from a DN file, and display verbose results every 5 seconds (`-j 5 -v`).

```
$ rsearch -h hostname -p port -s "" -f "" \
          -D "bindDN" -w password -B DNfile -c -t 2 -j 5 -v
```

Bind and Compare Rate

This command will create two threads (`-t 2`), each of which will bind before every compare operation (`-c -b -L`) and unbind afterwards, while displaying verbose results every 5 seconds (`-j 5 -v`). Binding will use a random DN from a DN file (`-B` *DNfile* `-x`) with the DN's corresponding UID as the password. Another DN will be randomly selected from the same file and used as the target of the compare operation based again on its UID. This implies that directory entries for DNs appearing in the *DNfile* must have the same UID and password.

```
$ rsearch -h hostname -p port -s "" -f "" -D "" -w "" \
    -B DNfile -x -c -b -L -t 2 -j 5 -v
```

NOTE In this form, the `-D ""` and `-w ""` options have no effect but are required on the command-line.

Measuring Modify Operations

The following examples will measure the performance of modify operations. The `rsearch` tool performs a modify operation on the `telephonenumber` and `description` attributes. These examples assume that the `telephonenumber` attribute is indexed in your test directory and that the `description` attribute is not.

In these examples, the `-s ""` and `-f ""` options have no effect but are required on the command-line. Also, adding the -x option will use the same DN file for both binding and modification. In this case however, all DNs in your DN file must have modification rights to all entries corresponding to those DNs.

Indexed Attribute Modify Rate

These commands will bind as the root DN (`-D "cn=Directory Manager" -w password`) and repeatedly perform a modify operation on the `telephonenumber` attribute of randomly chosen entries from the DN file (`-B` *DNfile* `-M`). While this example scenario assumes that the `telephonenumber` attribute is indexed, the command does not verify this and will function normally even if it is not.

```
$ rsearch -h hostname -p port -s "" -f "" \
          -D "cn=Directory Manager" -w password -B DNfile -M

$ rsearch -h hostname -p port -s "" -f "" \
          -D "cn=Directory Manager" -w password -B DNfile -M -b
```

The first form of this command will bind only once and measure the average rate of modify operations. The second form with the `-b` option will rebind at every modify operation to measure the average time for the bind and modify operation sequence.

Non-Indexed Attribute Modify Rate

These commands will bind as the root DN (`-D "cn=Directory Manager" -w password`) and repeatedly perform a modify operation on the `description` attribute of randomly chosen entries from the DN file (`-B` *DNfile* `-m`). While this example scenario assumes that the `description` attribute is not indexed, the command does not verify this and will function normally even if it is.

```
$ rsearch -h hostname -p port -s "" -f "" \
          -D "cn=Directory Manager" -w password -B DNfile -m
$ rsearch -h hostname -p port -s "" -f "" \
          -D "cn=Directory Manager" -w password -B DNfile -m -b
```

The first form of this command will bind only once and measure the average rate of modify operations. The second form with the `-b` option will rebind at every modify operation to measure the average time for the bind and modify operation sequence.

Measuring Delete Operations

The following examples will measure the performance of delete operations in your directory. The `-s ""` and `-f ""` options have no effect but are required on the command-line.

Delete Rate

This command will bind once as the root DN (`-D "cn=Directory Manager" -w` *password*) and repeatedly delete entries whose DN was randomly chosen from the DN file (`-B` *DNfile* `-d`).

```
$ rsearch -h hostname -p port -s "" -f "" \
          -D "cn=Directory Manager" -w password -B DNfile -d -T 7
```

TIP Because DNs are randomly chosen from the DN file, errors will occur when the tool tries to delete entries that have already been chosen and deleted. To minimize these errors, set the time limit option (`-T 7`) and use a DN file with a large number of DNs.

Bind and Delete Rate

This command will repeatedly perform delete operations on entries randomly chosen from a DN file (`-B` *DNfile* `-d`), while rebinding every time (`-b -L`) as the root DN (`-D "cn=Directory Manager" -w` *password*).

```
$ rsearch -h hostname -p port -s "" -f "" -D "cn=Directory Manager" \
          -w password -B DNfile -d -b -L -T 7
```

NOTE Do not use the `-x` option when performing bind and delete operations; this will cause `rsearch` to attempt to bind with DNs that may have already been deleted.

Part 4

LDAP v3 Tools

Chapter 13

The Search Performance Measurement Tool

The `searchrate` tool measures the performance of search operations in a Lightweight Directory Access Protocol (LDAP) v3 directory. This chapter provides instructions on how to use the `searchrate` tool. It contains the following sections:

- Overview
- Command Usage
- Sample Output
- Command-Line Examples

Overview

The `searchrate` tool measures the performance of search operations in an LDAP v3 directory. The DSRK and its updates include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory. Unlike the `rsearch` tool described in Chapter 12, it performs searches only.

NOTE As with all measures of performance, results depend upon many factors, such as the options and parameter values given, directory configuration, machine load, and network traffic, and should be analyzed accordingly. In order to obtain accurate performance measurements, both client and server machines should be dedicated to the performance test and properly tuned. (See Chapter 10, “The idsktune Optimization Tool” for tuning information.)

Command Usage

Using multiple threads, the `searchrate` tool simulates a search load on a directory server. Each thread performs LDAP bind and search operations repeatedly as quickly as possible, and the tool displays average results at regular intervals. The command-line options let you configure the binding sequence and the scope of the searches. The `searchrate` tool has the following built-in defaults:

- All operations use the LDAP v3 protocol. The tool cannot be used to test directories that only support LDAP v2.
- The tool uses simple or anonymous binding. No secure binding is possible.
- Referrals are never followed.
- The time and size limits for search results are not modifiable. The default time limit for a synchronous search is 10 seconds. All other default values are those defined by the directory server.

In general, when the `searchrate` tool encounters an error, it displays a message and continues running. It will attempt to bind again or search again indefinitely, even after encountering an error.

Syntax

The syntax of the `searchrate` tool on the command-line takes the following form:

```
searchrate -b "baseDN" -f "filter" [ options ]
```

Where:

- *baseDN* is the suffix, usually in double quotes ("") for the shell, and represents the subtree to be searched.
- *filter* is an RFC 2254-compliant LDAP search filter, usually in double quotes ("") for the shell. (See “LDAP Search Filters” in Chapter 4 of the *Sun ONE Directory Server Getting Started Guide*.)
- *options* are the command-line options and their parameters described in “Options.”

Options

The `searchrate` options and parameters are described in Table 13-1. Running the `searchrate` command on the command-line without any options or parameters will display the usage help text that briefly describes all options.

Table 13-1 Command-Line Options for `searchrate`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. The default is `localhost`.
`-p`	*port*	Specify the port number when accessing the directory server host. The default is 389.
`-D`	*bindDN*	Specify a bind DN for accessing the directory, usually in double quotes (`""`) for the shell. Depending on its access permissions, the bind DN may influence authentication and search performance. If the bind DN and its password are omitted, the tool will perform search operations without binding (as permitted by LDAP v3). For anonymous binding, you must explicitly specify an empty bind DN and password (`-D "" -w ""`).
`-w`	*password*	Specify the password for the bind DN.
`-b`	*baseDN*	Specify the base DN for the search operations, usually in double quotes (`""`) for the shell. See “Randomly Generated Numbers” on page 189 on how to use `%s` or `%d` placeholders for including random strings or numbers with the `-i` or `-r` options, respectively.
`-s`	*scope*	Specify the scope of a search with one of the following values: • `base` - For searching only the base entry. • `one` - For searching one level below the base entry. • `sub` - For searching all levels below the entry. This is the default.
`-f`	*filter*	Specify the filter for all search operations. See “Randomly Generated Numbers” on page 189 on how to use `%s` or `%d` placeholders for including random strings or numbers with the `-i` or `-r` options, respectively.
`-g`	*seconds*	Specify the time limit in seconds for each search operation. The default is 10 seconds. You may need to set the limit higher if you specify a broad search, such as `(cn=*)` over a large directory.
`-i`	*inputFile*	Give the name of the file containing strings that will be randomly substituted into `%s` placeholders in the base DN and filter. Each line of an input file is treated as a separate string. Use this option as many times as you have `%s` placeholders in the filter. See “Randomly Generated Numbers” on page 189 for more information.

Table 13-1 Command-Line Options for `searchrate` *(Continued)*

Option	Parameter	Purpose
`-r`	*maxRand*	Give the maximum range for random numbers to be substituted into `%d` placeholders in the base DN and filter. You may specify this option twice: the first random number will be in the range `[0,` *maxRand1*`-1]`, the second will be in the range `[1,` *maxRand2*`]`. See "Randomly Generated Numbers" on page 189 for more information.
`-A`	*attribute*	Specify an attribute to be retrieved during search operations. You may use this option any number of times on the command-line: only the specified attributes will be retrieved. When omitted, all searches will return all attributes for all matching entries. Whether or not this attribute is used, no attribute values are ever displayed. However, retrieving different attributes may influence search performance.
`-k`		Keep connections open between searches. With this option, the `searchrate` tool will measure only the execution time of the bind and search operations. When this option is omitted, the initialization and freeing of the connection are also measured as part of each search sequence.
`-K`		Keep connections and binds open between searches. With this option, the `searchrate` tool will measure only the execution time of the search operation. When this option is omitted, the initialization, binding, unbinding, and freeing of the connection is also measured as part of each search sequence.
`-u`		When used with the `-D -w` options, specify that the tool should not unbind from the sever, just close the socket for the connection. This option has no effect when either `-k` or `-K` options are specified.
`-a`		Specify asynchronous search mode. With this option, the `searchrate` tool will not wait for search results before starting the next search. As a result, performance measurements will reflect `searchrate` execution speed, not directory server search speed. This option cannot be used when the `-D` and `-w` options are given.
`-t`	*threads*	Specify the number of threads that `searchrate` will run in parallel. The output displays the average performance of all threads combined. The default is a single thread.
`-j`	*seconds*	Specify the measurement and display interval, in seconds; the default is 5. `searchrate` repeats the search sequence as many times as possible during the interval and prints results after each interval elapses.
`-m`	*searchOps*	Specify the maximum, cumulated number of search operations for each thread to perform. When this option is not specified, all threads will repeat the search sequence indefinitely.
`-q`		Quiet output mode: the measurements for each interval will not be displayed.
none		Display the usage help text that briefly describes all options.

Randomly Generated Numbers

One concern for accurate performance measurements is to simulate real usage conditions and reduce any artifacts due to the repetitive nature of the tests. For example, when the same entry is retrieved for every search, it will be cached by the directory server and returned without performing a full search. Under normal usage conditions, only a certain percentage of searches will be resolved quickly through the entry cache, so the test results will be skewed.

Syntax

To simulate real usage conditions, the `searchrate` tool has a mechanism for randomizing both the base DN and the search filter used for each search. Both the base DN and filter strings may use the following syntax:

- `%d` includes random numbers up to the value defined by the `-r` option.
- `%s` includes random strings from the files defined by the `-i` option.

To include either randomly generated numbers or random strings from an input file, specify the following placeholders in the base DN, in the filter parameters, or in both on the command-line:

- The first occurrence of `%d` will be replaced by a random number in the range `[0,` *maxRand1*`-1]`, where *maxRand1* is defined by the first occurrence of the `-r` option on the command-line.
- The second occurrence of `%d` will be replaced by a random number in the range `[1,` *maxRand2*`]`, where *maxRand2* is defined by the second occurrence of the `-r` option on the command-line.
- The `%s` placeholders will be replaced by a random string from the input file defined by the `-i` options. The first placeholder in each of the base DN and the filter will be replaced by a random string from the input file specified by the first `-i` option, the second placeholder in each by a random string from the second file, and so on. Each line of an input file is treated as a complete string.

Substitution Rules

The tool applies the following rules for substitutions. An incorrect command-line will return a usage error.

- You must specify at least as many `-r` options as `%d` placeholders you use, and as many `-i` options as `%s` placeholders.

- Placeholder substitution will occur only in the base DN and search filter parameters. To use the literal strings "%d" and "%s" within these parameters, you must use "%%d" and "%%s", respectively.
- Within a DN or filter, you may use only one type of placeholder. However, each parameter may use a different type.
- When both parameters specify the same type of placeholder, the same value or values will be substituted in both for a given search.

Input File

The input file is a plain text file that contains the strings for substitution. Each line, including any whitespace, is taken to be one string. File contents should be adapted to the intended substitutions: either in the base DN, in the search filter, or in both. For example, the following input file could be used with the option `-f "sn%s"` for performing different searches on surnames:

```
=C*
=Jones
=Smith
>=Jones
<=Jones
~=Turner
```

The size and contents of the file should be adapted to the directory that you are testing. Generally, the type and number of different searches that can be chosen randomly from the input file should resemble the expected usage of your directory. For example, to test the performance of complex searches, use several input files with a compound filter string such as `"(|(cn=%s)(cn=%s))"`.

Sample Output

Code Example 13-1 is sample output that might be retreived from the `searchrate` tool. When running, the `searchrate` tool displays one line of measurements every interval (5 seconds by default). All data on an output line concerns only the elapsed interval. Use the `-j` *seconds* option to specify a different interval length.

Code Example 13-1 Sample Output From searchrate Tool

```
$ searchrate -h hostname -b "dc=example,dc=com" -f "cn=a*" -t 4
Avg r=  74.75/thr ( 59.80/sec), total=   299
Avg r=  76.00/thr ( 60.80/sec), total=   304
Avg r=  74.50/thr ( 59.60/sec), total=   298
```

Code Example 13-1 Sample Output From searchrate Tool *(Continued)*

```
Avg r=  56.00/thr ( 44.80/sec), total=   224
Avg r=  73.50/thr ( 58.80/sec), total=   294
Avg r=  75.25/thr ( 60.20/sec), total=   301
^C
```

Reading an output line backwards, it shows:

- The total number of search operations completed by all threads during the full interval.
- The rate in parentheses is the average number of searches per second for all threads (the total divided by the number of seconds in the interval).
- The given Avg is the average number of operations per thread during the interval (the total divided by the number of threads).

Command-Line Examples

The examples in this section will measure bind and search performance in your directory server in various scenarios. Results will be meaningful only if factors such as machine load and network traffic remain constant during and between tests.

In these examples, you will need to provide the base DN and filter strings for valid entries in your test directory. You will also need to adapt them to your environment:

- The *hostname* and *port* placeholders should be replaced with the hostname and port number of your directory. The *baseDN* should represent the root of the directory or subtree you wish to test.
- For realistic results, thread numbers and time limit options should be scaled according to your directory's expected load.
- You will need to provide input files that correspond to the contents of your directory. See "Randomly Generated Numbers" on page 189 for more information.

Simple Search

```
$ searchrate -h hostname -p port -b "dc=example,dc=com" \
             -f "cn=*john*" -t 3 -j 60
```

This command will launch 3 threads (`-t 3`), each of which will repeatedly open a connection(but not bind), search for entries under the `"dc=example,dc=com"` suffix that contain the `"cn=*john*"` substring, and close the connection (no `-D -w -k -K` options). The tool will display combined results for all threads at one minute intervals (`-j 60`).

Search Using an Input File

Because every search uses the same filter and returns the same entries, entry caching in the directory server will influence results. In all of the following examples, the use of an input file containing filter strings helps provide more realistic measurements. During each iteration, the tool will search for a different, random surname taken from the input file such as the one in “Randomly Generated Numbers” on page 189 (`-f "sn%s" -i` *inputFile*).

NOTE Input files might also contain different kinds of filter strings that could give different performance results:

- Strings containing wildcards will give measurements that reflect substring search speed.
- Strings without wildcards will give measurements that reflect the search speed for exact matches.
- Filter strings may also include operators other than equality, such as ranges that are greater than (>=) or less than (<=) a given value, or approximate spelling searches (~=).

Open, Bind, and Search Rate

This first command will use a single thread to repeatedly open a connection, bind with the given credentials (`-D "`*bindDN*`" -w` *password*), perform a search under the `"dc=example,dc=com"` suffix and then unbind. The unbinding operation also closes the connection.

```
$ searchrate -h hostname -p port -b "dc=example,dc=com" \
             -D "bindDN" -w password -f "sn%s" -i inputFile
```

This second command is similar to the previous one, except that at each iteration, the connection and binding will simply be dropped by closing the socket (`-u`). This behavior is allowed by the LDAP protocol, and this test verifies that connections that are not unbound are properly handled by the directory server.

```
$ searchrate -h hostname -p port -b "dc=example,dc=com" \
             -D "bindDN" -w password -u -f "sn%s" -i inputFile
```

NOTE Both of the commands may also use anonymous binding (`-D "" -w ""`) that may give different performance results due to handling anonymous access permissions.

Bind and Search Rate

This command will use a single thread to keep a connection open (`-k`) to repeatedly bind with the given credentials and perform a search under the `"dc=example,dc=com"` suffix. The LDAP protocol allows clients to bind multiple times without unbinding, and this test measures performance in this situation.

```
$ searchrate -h hostname -p port -b "dc=example,dc=com" \
             -D "bindDN" -w password -k -f "sn%s" -i inputFile
```

NOTE You may also use anonymous binding (`-D "" -w ""`) to test the performance results using anonymous access.

Search Rate Alone

This command will use a single thread to keep the connection and the bind open (`-K`) and repeatedly perform a search under the `"dc=example,dc=com"` suffix. This will isolate the performance measurements of the search operation alone. You may also use anonymous binding (`-D "" -w ""`) to test the performance results using anonymous access.

```
$ searchrate -h hostname -p port -b "dc=example,dc=com" \
             -D "bindDN" -w password -K -f "sn%s" -i inputFile
```

In this command, the single thread will only open a connection without binding (`-k` but no `-D -w`) and repeatedly perform search operations. Unbound search operations are allowed by the LDAP v3 protocol, and this test isolates the performance of the search operation alone in this situation.

```
$ searchrate -h hostname -p port -b "dc=example,dc=com" \
             -k -f "sn%s" -i inputFile
```

Complex Searches

Searches that use compound filters having multiple terms are common and take longer to perform. To create realistic tests of complex search performance, use several placeholders in the filter string and specify multiple input files.

```
$ searchrate -h hostname -p port -b "dc=example,dc=com" \
             -k -f "(|(cn=%s)(cn=%s)))" -i cnFile -i cnFile
```

Note that you must specify as many `-i` options as `%s` placeholders, even if the input filename is the same for both.

Chapter 14

The Modify Performance Measurement Tool

The `modrate` tool measures the performance of modify operations in a Lightweight Directory Access Protocol (LDAP) v3 directory. This chapter provides instructions on how to use the `modrate` tool. It contains the following sections:

- Overview
- Command Usage
- Sample Output
- Command-Line Examples

Overview

The `modrate` tool measures the performance of modify operations in an LDAP v3 directory. It is similar to the `rsearch` functionality described Chapter 12, "The rsearch Search Tool," except that it performs modifications on random user-defined attributes. The DSRK and its updates include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory.

NOTE As with all measures of performance, results depend upon many factors, such as the options and parameter values given, directory configuration, machine load, and network traffic, and should be analyzed accordingly. In order to obtain accurate performance measurements, both client and server machines should be dedicated to the performance test and properly tuned. (See Chapter 10, "The idsktune Optimization Tool" for tuning information.)

Command Usage

Using multiple threads, the `modrate` tool repeatedly performs modify operations on a directory server. Threads may be configured to open connections and perform LDAP bind operations with every modification. The command-line options let you specify the target entries and attributes to be modified. The `modrate` tool has the following built-in defaults:

- All operations use the LDAP v3 protocol. The tool cannot be used to test directories that only support LDAP v2.
- The tool uses simple or anonymous binding. No secure binding is possible.
- Referrals are never followed.
- The time limit for operations is not modifiable. The default time limit is that defined by the directory server.

The tool displays performance results at regular intervals. In general, when the `modrate` tool encounters an error, it displays a message and continues running. It will attempt to bind again or modify again indefinitely, even after encountering an error.

Syntax

The syntax of the `modrate` tool on the command-line takes the following form:

```
modrate -D "bindDN" -w password -b "baseDN" [ options ] \
        -M "attribute:length:charSet" ...
```

Where:

- *bindDN* and *password* are bind credentials with write permission to the target entry or entries. The bind DN is usually in double quotes ("") for the shell.
- *baseDN* is the DN of the entry to be modified, usually in double quotes ("") for the shell. The target entry should support the attributes given by the `-M` option, either for modification or for addition when not already present. The base DN may use either of the following placeholders:
  - `%d` to include random numbers up to the value given by the `-r` option.
  - `%s` to include a random string from the file given with the `-i` option.

NOTE See “Randomly Generated Target Entries” on page 199 for more information.

- *options* are the command-line options and their parameters described in "Options."
- The `-M` parameter gives the information needed to generate random attribute values. There can be any number of `-M` parameters on the command-line.
  - *attribute* is the name of an existing attribute of the target entry or entries given by the base DN. If this attribute does not exist on the target entry, it will be added and be subject to schema checking if it is enabled in the directory.
  - *length* is an integer giving the desired number of characters in random attribute string values.
  - *charSet* specifies a set of individual ASCII characters (*c*) and ranges of characters, according to the following syntax:

    (*c**([*c*-*c*])*)* for example: `[A-Z][a-z][0-9]`

  For each modify operation, a `modrate` thread will randomly choose one of the *attribute* names and generate a new string value of the given *length* by choosing each character randomly from the given *charSet*.

Options

The `modrate` options and parameters are described in Table 14-1. Running the `modrate` command on the command-line without any options or parameters will display brief descriptions of all the command-line options.

Table 14-1 Command-Line Options for `modrate`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. The default is `localhost`.
`-p`	*port*	Specify the port number when accessing the directory server host. The default is 389.
`-D`	*bindDN*	Specify a bind DN for accessing the directory, usually in double quotes ("") for the shell. Depending on its write permissions, the bind DN may influence authentication and modify performance.
`-w`	*password*	Specify the password for the bind DN.
`-b`	*baseDN*	Specify the base DN of the target entry, usually in double quotes ("") for the shell. See "Randomly Generated Target Entries" on page 199 on how to include %s or %d placeholders for random strings or numbers with the `-i` or `-r` options, respectively.

Table 14-1 Command-Line Options for `modrate` *(Continued)*

Option	Parameter	Purpose
`-M`	*modString*	Specify the name of an attribute to modify and how to randomly generate a new value for it. The *modString* format (*attribute*:*length*:*charSet*) is described in "Syntax" on page 196.
`-i`	*inputFile*	Give the name of the file containing strings that will be randomly substituted into `%s` placeholders in the base DN. Each line of the input file will be treated as a separate string. See "Randomly Generated Target Entries" on page 199 for more information. **CAUTION: This option is incompatible with the `-r` option.**
`-r`	*maxRand*	Give the maximum range for random numbers to be substituted into `%d` placeholders in the base DN. You may specify this option twice: the first random number will be in the range `[0,` *maxRand1*`-1]`, the second will be in the range `[1,` *maxRand2*`]`. See "Randomly Generated Target Entries" on page 199 for more information. **CAUTION: This option is incompatible with the `-i` option.**
`-k`		Keep connections open between modify operations. With this option, the `modrate` tool will measure only the execution time of bind and modify operations. When this option is omitted, the initialization and freeing of the connection is also measured as part of each modification sequence.
`-K`		Keep connections and binds open between modify operations. With this option, the `modrate` tool will measure only the execution time of modify operations. When this option is omitted, the initialization, binding, unbinding, and freeing of the connection is also measured as part of each sequence.
`-u`		Specify that the tool should not unbind from the server and just close the socket for the connection. This option has no effect when either `-k` or `-K` options are specified.
`-O`	*hopLimit*	(Capital letter O) Specify the maximum number of referral hops to follow while finding an entry to modify. By default, there is no limit.
`-R`		Specify that referrals should *not* be followed. By default, referrals are followed automatically.
`-a`		Specify asynchronous modification mode. This option is currently not working.
`-t`	*threads*	Specify the number of threads that `modrate` will run in parallel. The output displays the average performance of all threads combined. The default is a single thread.
`-j`	*seconds*	Specify the measurement and display interval, in seconds; the default is 5. `modrate` repeats the modify operation sequence as many times as possible during the interval and prints results after each interval elapses.
`-m`	*modifyops*	Specify the maximum, cumulated number of modify operations for each thread to perform. When this option is not specified, all threads will repeat the modify sequence indefinitely.

Table 14-1 Command-Line Options for `modrate` *(Continued)*

Option	Parameter	Purpose
`-q`		Quiet output mode: the measurements for each interval will not be displayed.
none		Display the usage help text that briefly describes all options.

Randomly Generated Target Entries

To simulate real usage conditions and reduce artifacts due to the repetitive nature of the tests, the `modrate` tool provides a mechanism for choosing a random target entry.

Syntax

To simulate real usage conditions, the `modrate` tool has a mechanism for generating random numbers or random strings from an input file. To do this, specify the following placeholders in the base DN parameter on the command-line:

- The first occurrence of `%d` will be replaced by a random number in the range `[0,` *maxRand1*`-1]`, where *maxRand1* is given by the first occurrence of the `-r` option on the command-line.
- The second occurrence of `%d` will be replaced by a random number in the range `[1,` *maxRand2*`]`, where *maxRand2* is given by the second occurrence of the `-r` option on the command-line.
- The `%s` placeholder will be replaced by a random string from an *inputFile* given by the `-i` option. Each line of this file is treated as a complete string to insert.

Substitution Rules

The tool applies the following rules for substitutions. An incorrect command-line will return a usage error:

- You must specify at least as many `-r` options as `%d` placeholders you use.
- Placeholder substitution will occur only in the base DN parameters. To use the literal strings "`%d`" and "`%s`" within this parameter, you must use "`%%d`" and "`%%s`", respectively.
- You may use only one type of placeholder, either decimal number or string.

Input File

The input file is a plain text file that contains the strings for substitution. Each line, including any whitespace, is taken to be one string. Depending on how the placeholder is used, the file may contain full or partial DNs. For example, the `-b "uid=%s,ou=people,dc=example,dc=com"` option could be used with the input file detailed in Code Example 14-1 on page 200.

Code Example 14-1 Sample Input File for modrate Tool

```
bjensen
bjense2
mtalbot
kcarter
svaughan
pshelton
```

The size and contents of the file should be adapted to the directory that you are testing. Generally, the number of possible target entries should be as large as possible to ensure performance is being measured across the whole directory.

Sample Output

Code Example 14-2 is sample output that might be retreived from the `modrate` tool. When running, the `modrate` tool displays one line of measurements every interval (5 seconds by default). All data on an output line concerns only the elapsed interval. Use the `-j` *seconds* option to specify a different interval length.

Code Example 14-2 Sample Output From modrate Tool

```
Avg r=  37.00/thr (  7.40/sec), total=    37
Avg r=  36.00/thr (  7.20/sec), total=    36
Avg r=  29.00/thr (  5.80/sec), total=    29
Avg r=  35.00/thr (  7.00/sec), total=    35
Avg r=  38.00/thr (  7.60/sec), total=    38
Avg r=  39.00/thr (  7.80/sec), total=    39
^C
```

Reading an output line backwards, it shows:

- The total number of modify operations completed by all threads during the full interval.
- The rate in parentheses is the average number of modify operations per second for all threads (the total divided by the number of seconds in the interval).
- The given Avg is the average number of operations per thread during the interval (the total divided by the number of threads).

Command-Line Examples

The examples in this section measure bind and modify performance in your directory server in various scenarios. Results are meaningful only if factors such as machine load and network traffic remain constant during and between tests.

In these examples, you should provide DNs for modifiable entries in your test directory. You will need to adapt these examples to your environment:

- The *hostname* and *port* placeholders should be replaced with the hostname and port number of your directory.
- For realistic results, thread numbers should be scaled according to your directory's expected load.
- You should also specify modifiable attributes in the target entries with character sets that conform to your directory's schema.

Open, Bind and Modify Rate

The following command will launch 3 threads (`-t 3`), each of which will repeatedly open a connection, bind with the given credentials (`-D "`*bindDN*`" -w` *password*), modify the `telephonenumber` attribute of the `bjensen` entry, unbind, and close the connection (no `-u -k -K` options). The new attribute value for each modification is a sequence of 10 random digits (`-M telephonenumber:10:[0-9]`) that simulate a telephone number. The tool will display combined results for all threads at one minute intervals (`-j 60`).

```
$ modrate -h hostname -p port -D "bindDN" -w password \
          -b "uid=bjensen,ou=people,dc=example,dc=com" \
          -M 'telephonenumber:10:[0-9]' -t 3 -j 60
```

Adding the `-u` option to this command-line will test whether the directory server handles clients that don't unbind before disconnecting.

Searches with an Input File

Because every modify operation applies to the same entry, entry caching in the directory server will influence results. In the following examples, the use of an input file containing UID strings helps provide more realistic measurements. (See “Randomly Generated Target Entries” on page 199 for more information.)

Bind and Modify Rate

The following command will use a single thread to keep a connection open (`-k`) to repeatedly bind with the given credentials and perform a modify operation on an entry whose UID is randomly chosen from the *inputFile*. The LDAP protocol allows clients to bind multiple times without unbinding, and this test measures performance in this situation.

```
$ modrate -h hostname -p port -D "bindDN" -w password \
          -b "uid=%s,ou=people,dc=example,dc=com" -i inputFile \
          -k -M 'telephonenumber:10:[0-9]'
```

Modify Rate Alone

The following command will use a single thread to keep the connection and the bind open (`-K`) and repeatedly perform a modify operation on an entry whose UID is randomly chosen from the *inputFile* at each iteration. This will isolate the performance measurements of the modify operation alone.

```
$ modrate -h hostname -p port -D "bindDN" -w password \
          -b "uid=%s,ou=people,dc=example,dc=com" -i inputFile \
          -K -M 'telephonenumber:10:[0-9]'
```

Chapter 15

The Rate of Authentication Measurement Tool

The `authrate` tool measures the possible rate of authentication to an LDAP v3 directory. This chapter provides instructions on how to use the `modrate` tool. It contains the following sections:

- Overview
- Command Usage
- Sample Output
- Command-Line Examples

Overview

The `authrate` tool measures the possible rate of authentication to an LDAP v3 directory. It is similar to the `rsearch` functionality described in Chapter 12, "The rsearch Search Tool," providing a mechanism for using random bind DN and password credentials. The DSRK and its updates include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory.

NOTE As with all measures of performance, results depend upon many factors, such as the options and parameter values given, directory configuration, machine load, and network traffic, and should be analyzed accordingly. In order to obtain accurate performance measurements, both client and server machines should be dedicated to the performance test and properly tuned. (See Chapter 10, "The idsktune Optimization Tool" for more information.)

Command Usage

Using multiple threads, the `authrate` tool repeatedly initializes a connection and binds to a directory server, without performing any other operation. Threads may be configured to keep open connections and perform LDAP binds repeatedly. The command-line options let you specify the bind credentials. The `modrate` tool has the following built-in defaults:

- All operations use the LDAP v3 protocol. The tool cannot be used to test directories that only support LDAP v2.
- The tool uses simple or anonymous binding. No secure binding is possible.

The tool displays performance results at regular intervals. In general, when the `authrate` tool encounters an error, it displays a message and continues running. It will attempt to bind again indefinitely, even after encountering an error.

Syntax

The syntax of the `authrate` tool on the command-line takes the following form:

```
authrate -D "bindDN" -w password [ options ]
```

Where:

- *bindDN* and *password* are the bind credentials, with the bind DN is usually in double quotes ("") for the shell. The bind DN and password may use `%s` and `%d` placeholders to include random strings and numbers. (See "Randomly Generated Bind DNs" on page 206 for more information.)
- *options* are the command-line options and their parameters described in "Options."

Options

The `authrate` options and parameters are described in Table 15-1. The `authrate -H` command and option when run on the command-line will display brief descriptions of all the command-line options.

Table 15-1 Command-Line Options for `authrate`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. The default is `localhost`.

Table 15-1 Command-Line Options for `authrate` *(Continued)*

Option	Parameter	Purpose
`-p`	*port*	Specify the port number when accessing the directory server host. The default is 389.
`-D`	*bindDN*	Specify a bind DN for accessing the directory, usually in double quotes (`""`) for the shell. See “Randomly Generated Bind DNs” on page 206 for information on how to include `%s` or `%d` placeholders for random strings or numbers using the `-i` or `-r` option, respectively.
`-w`	*password*	Specify the password for the bind DN. The password may also contain `%s` or `%d` placeholders that will use the same random strings or numbers as the bind DN substitutions. See “Randomly Generated Bind DNs” on page 206 for details.
`-i`	*inputFile*	Give the name of the file containing strings that will be randomly substituted into `%s` placeholders in the bind DN and password. Each line of the input file is treated as a separate string. See “Randomly Generated Bind DNs” on page 206 for more information.
`-r`	*maxRand*	Give the maximum range for random numbers to be substituted into `%d` placeholders in the bind DN and password. You may specify this option twice: the first random number will be in the range `[0,` *maxRand1*`-1]`, the second will be in the range `[1,` *maxRand2*`]`.
`-k`		Keep connections open when performing binds. With this option, the `authrate` tool will measure only the execution time of the bind operation. When this option is omitted, the initialization and freeing of the connection is also measured as part of each authentication sequence.
`-u`		Specify that the tool should not unbind from the server and just close the socket for the connection. This option has no effect when the `-k` option is specified.
`-t`	*threads*	Specify the number of threads that `authrate` will run in parallel. The output displays the average performance of all threads combined. The default is a single thread.
`-j`	*seconds*	Specify the measurement and display interval, in seconds; the default is 5. `authrate` repeats the authentication sequence as many times as possible during the interval and prints results after each interval elapses.
`-m`	*bindOps*	Specify the maximum number of bind operations for each thread to perform. When this option is not specified, all threads will repeat the authentication sequence indefinitely.
`-q`		Quiet output mode: the measurements for each interval will not be displayed.
`-H`		Display the usage help text that briefly describes all options.

Randomly Generated Bind DNs

To simulate real usage conditions and reduce any artifacts due to the repetitive nature of the tests, the `authrate` tool provides a mechanism for generating a random bind DN for authentication.

Syntax

You can include randomly generated numbers by specifying the following placeholders:

- In the bind DN, the first and second occurrences of `%d` will be replaced by a random number in the ranges `[0, `*maxRand1*`-1]` and `[1, `*maxRand2*`]`, respectively, where *maxRand1* and *maxRand2* are given by the first and second occurrences of the `-r` option on the command-line. The bind DN parameter may have no more than two `%d` placeholders.
- In the password parameter, all occurrences of `%d` will be replaced by the same random number in the range `[0, `*maxRand1*`-1]`, where *maxRand1* is given by the first occurrence of the `-r` option on the command-line. The password parameter may have up to 8 placeholders, to generate a password with enough characters when the random substitution is a single digit.
- In both the bind DN and password parameter, the `%s` placeholder will be replaced by the same random string from an input file given by the `-i` option. Each line of this file is treated as a complete string to insert.

Substitution Rules

The tool applies the following rules for substitutions. An incorrect command-line will return a usage error:

- You must specify at least as many `-r` options as `%d` placeholders you use in the bind DN.
- To use the literal strings "`%d`" and "`%s`" within the bind DN or password, you must use "`%%d`" and "`%%s`", respectively.
- You may use only one type of placeholder, either decimal number or string.

To use the random authentication, your test directory must contain entries written with these substitution rules in mind. Because the same random number or string will be substituted into both bind DN and password, your entries must have matched DN and password pairs. For example, the following entries have one number in the DN and two in the password:

```
dn: cn=test0,dc=example,dc=com
password: auth00

dn: cn=test1,dc=example,dc=com
password: auth11

dn: cn=test2,dc=example,dc=com
password: auth22
...

dn: cn=test10,dc=example,dc=com
password: auth1010
...

dn: cn=test99,dc=example,dc=com
password: auth9999
```

The following command will test authentication using these entries:

```
authrate -D "cn=test%d,dc=example,dc=com" -w "auth%d%d" -r 100
```

Sample Output

Code Example 15-1 is sample output that might be retreived from the `authrate` tool. When running, the `authrate` tool displays one line of measurements every interval (5 seconds by default). All data on an output line concerns only the elapsed interval. Use the `-j` *seconds* option to specify a different interval length.

Code Example 15-1 Sample Output From authrate

```
Avg r= 754.00/thr (150.80/sec), total=   754
Avg r= 774.00/thr (154.80/sec), total=   774
Avg r= 829.00/thr (165.80/sec), total=   829
Avg r= 825.00/thr (165.00/sec), total=   825
Avg r= 836.00/thr (167.20/sec), total=   836
Avg r= 837.00/thr (167.40/sec), total=   837
^C
```

Reading an output line backwards, it shows:

- The total number of authentications completed by all threads during the full interval.

- The rate in parentheses is the average number of authentications per second for all threads (the total divided by the number of seconds in the interval).
- The given Avg is the average number of authentications per thread during the interval (the total divided by the number of threads).

Command-Line Examples

The examples in this section will measure authentication performance in your directory server in various scenarios. Results will be meaningful only if factors such as machine load and network traffic remain constant during and between tests. These examples suppose the contents of your test directory are configured as detailed in “Randomly Generated Bind DNs” on page 206. You will need to adapt other parameters to your environment:

- The *hostname* and *port* placeholders should be replaced with the hostname and port number of your directory.
- For realistic results, thread numbers should be scaled according to your directory’s expected load.

Open and Bind Rate

This command will launch 3 threads (`-t 3`), each of which will repeatedly open a connection, bind with randomly generated credentials (`-D "cn=test%d,dc=example,dc=com" -w "auth%d%d" -r 100`), unbind, and close the connection (no `-u -k` options). The tool will display combined results for all threads at one minute intervals (`-j 60`).

```
$ authrate -h hostname -p port -t 3 -j 60 \
           -D "cn=test%d,dc=example,dc=com" -w "auth%d%d" -r 100
```

Adding the `-u` option to this command-line will test whether the directory server handles clients that don’t unbind before disconnecting.

Bind Rate Alone

This command will use a single thread to keep a connection open (`-k`) in order to repeatedly bind with randomly generated credentials (`-D "cn=test%d,dc=example,dc=com" -w "auth%d%d" -r 100`). The LDAP protocol allows clients to bind multiple times without unbinding, and this test measures performance in this situation.

```
$ authrate -h hostname -p port -k \
           -D "cn=test%d,dc=example,dc=com" -w "auth%d%d" -r 100
```

The Add Performance Measurement Tool

The `infadd` tool measures the performance of add operations in an LDAP v3 directory. This chapter provides instructions on how to use the `infaddd` tool. It contains the following sections:

- Overview
- Command Usage
- Command-Line Examples

Overview

The `infadd` (infinite add) tool measures the performance of entry add operations in an LDAP v3 directory. The DSRK and its updates include the latest version of the tool in the *DSRK_base*`/bin/dsrk52` directory. It generates entries containing random attribute values and adds them to the directory under a given suffix, performing operations continuously and computing an average operation rate at regular intervals.

NOTE As with all measures of performance, results depend upon many factors, such as the options and parameter values given, directory configuration, machine load, and network traffic, and should be analyzed accordingly. In order to obtain accurate performance measurements, both client and server machines should be dedicated to the performance test and properly tuned. (See Chapter 10, “The idsktune Optimization Tool” for more information.)

Command Usage

Using multiple threads, the `infadd` tool binds to a directory server and repeatedly performs LDAP add operations. All entries are added to the same subtree, one level below the *suffix* given on the command-line. New entries belong to the `inetOrgPerson` object class and have the attributes detailed in Code Example 16-1.

Code Example 16-1 Attributes of Added Entries

```
dn: cn= givenname sn UID, suffix
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: givenname sn UID
givenname: givenname
sn: sn
uid: UID
mail: givennameUID@siroe.com
telephonenumber: telephonenumber
userpassword: UID
audio: binaryData
```

The *givenname* and *sn* are randomly chosen either from data files containing names or from randomly generated strings. The following data files are provided in the `DSRK_base/data` subdirectory of the Directory Server Resource Kit:

- `dbgen-FamilyNames` contains over 13400 plausible surnames.
- `dbgen-GivenNames` contains over 8600 plausible first names.

The *UID* is a sequential numbering of new entries (see the `-I` option), and the *telephonenumber* is a random, US-format telephone number. The binary `audio` attribute is optional and can be used to test the addition of large entries (see the `-z` option).

NOTE The tool displays performance results at regular intervals. Performance of add operations is highly dependent upon whether or not the directory performs schema checking. Be sure your directory is configured so that the test scenario most closely resembles its actual usage.

Syntax

The syntax of the `infadd` tool on the command-line takes the following form:

```
infadd -s "suffix" -u "bindDN" -w password [ options ]
```

Where:

- *suffix* is the base DN under which all entries will be added.
- *bindDN* and *password* are the bind credentials, with the bind DN is usually in double quotes ("") for the shell.
- *options* are the command-line options and their parameters described in "Options."

Options

The `infadd` options and parameters are described in Table 16-1. Running the `infadd` command without any options or parameters will display the usage help text that briefly describes all options.

Table 16-1 Command-Line Options for `infadd`

Option	Parameter	Purpose
`-h`	*hostname*	Specify the hostname of the directory server. The default is `localhost`.
`-p`	*port*	Specify the port number when accessing the directory server host. The default is 389.
`-D`	*bindDN*	Specify a bind DN for accessing the directory, usually in double quotes ("") for the shell. The bind DN should have write permission in the subtree given by the `-s` *suffix* parameter.
`-w`	*password*	Specify the password for the bind DN.
`-B`	*baseDN*	Specify the base DN to use for all new entries, usually in double quotes ("") for the shell. This is effectively the common suffix for all entries to be added.
`-y`		Set the `TCP_NODELAY` mode on the TCP socket.
`-t`	*threads*	Specify the number of threads that `infadd` will run in parallel. Use the `-v` option for verbose output including measurements from each thread. The default is a single thread.
`-j`	*seconds*	Specify the measurement and display interval, in seconds; the default is `10`. `infadd` creates as many new entries as possible during the interval and prints results after each interval elapses.

Table 16-1 Command-Line Options for `infadd` *(Continued)*

Option	Parameter	Purpose
`-m`	*addOps*	(Solaris and UNIX only) Specify the approximate number of total add operations for the tool to perform. The tool will stop after the measurement interval where the total number of operations for all threads exceeds this parameter. When this option is not specified, all threads will continue adding entries indefinitely.
`-q`		Quiet output mode: the measurements for each interval will not be displayed.
`-v`		Verbose output at every interval: gives the measurements from each thread, including the minimum and maximum operation times observed, as well as the average over all threads.
`-I`	*startID*	(Solaris and UNIX only) For guaranteeing uniqueness of DNs, the `infadd` tool generates a sequential ID number, beginning with *startID*. This ID number is appended to the `cn` attribute and also used as the `uid` attribute of new entries. When this option is omitted, ID numbers begin at zero.
`-R`	*number*	(Solaris and UNIX only) Use randomly generated names in new entries. With this option, `infadd` will first generate the given *number* of random given names and surnames and then randomly select one of each when adding entries. A random name is a sequence of 7 to 12 random letters. When this option is omitted, the tool will use the contents of the `dbgen-GivenNames` and `dbgen-FamilyNames` files in the data subdirectory of the Sun ONE DSRK installation directory.
`-z`	*maxSize*	Specify that all new entries contain an `audio` attribute with a random binary value. The attribute's value is a set of randomly generated bytes, and the number of bytes is randomly chosen in the range [0, *maxSize*].
none		Display the usage help text that briefly describes all options.

Command-Line Examples

The examples in this section will measure entry addition performance in your directory server in various scenarios. These examples include sample output. Results will be meaningful only if factors such as machine load and network traffic remain constant during and between tests.

- The *hostname* and *port* placeholders should be replaced with the hostname and port number of your directory.
- For realistic results, thread numbers should be scaled according to your directory's expected load.

Multithreaded Verbose Output

The following command launches 3 threads (`-t 3`), each of which binds with the given credentials (`-D "`*bindDN*`" -w` *password*) and adds entries under the `"ou=people,dc=example,dc=com"` branch. The name attributes in the new entries are randomly generated from the `Given-Names` and `Family-Names` data files (no `-R` option). The verbose output (`-v`) gives the measured performance of each thread every 3 seconds (`-J 3`) until a total of at least 20 new entries have been added (`-M 20`).

```
$ infadd -h hostname -p port -D "bindDN" -w password \
          -B "ou=people,dc=example,dc=com" -t 3 -J 3 -M 20 -v
```

Code Example 16-2 details the sample output. The thread reports the following information:

- The minimum and maximum operation times during the elapsed interval.
- The number of operations performed during the elapsed interval.
- The total number of operations performed by that thread so far.

Because multiple threads are used, the average operation rate per thread and the total number of operations is also displayed after every interval. Finally, because of the operation limit option, the summary line is displayed at the end.

Code Example 16-2 Verbose Output of infadd Tool

```
Loading Given-Names ...
Loading Family-Names ...
infadd: 3 threads launched.
T1 min:  610ms, max: 1592ms, count:   3, total: 3
T2 min:  702ms, max: 1770ms, count:   2, total: 2
T3 min:  649ms, max: 1308ms, count:   3, total: 3
Average rate:   2.67, total: 8
T1 min:  513ms, max:  607ms, count:   4, total: 7
T2 min:  510ms, max:  655ms, count:   5, total: 7
T3 min:  533ms, max:  721ms, count:   5, total: 8
Average rate:   4.67, total: 22
Total added: 22, Avg rate: 7.33/thrd, 3.67/sec = 272.7msec/op
```

Random Strings and Binary Values

This command launches a single thread (no `-t` option) that binds with the given credentials (`-D "`*bindDN*`" -w` *password*) and adds entries under the `"ou=people,dc=example,dc=com"` branch. The name attributes in the new entries are chosen from 100 randomly generated given names and 100 randomly generated surnames (`-R 100`). The new entries also include the `audio` attribute with binary values up to 10KB long (`-z 10240`). The verbose output (`-v`) gives detailed measurements for the thread every 3 seconds (`-J 3`) until a total of at least 20 new entries have been added (`-M 20`). The data values in the output are the same as described in the previous example.

```
$ infadd -h hostname -p port -D "bindDN" -w password \
         -B "ou=people,dc=example,dc=com" \
          -R 100 -z 10240 -J 3 -M 20 -v
```

Code Example 16-3 details the sample output.

Code Example 16-3 Sample Output From infadd Tool

```
Generating random names: 100.  Done.
Generating random names: 100.  Done.
infadd: 1 thread launched.
T1 min:  123ms, max:  241ms, count:   8, total: 8
T1 min:  133ms, max: 2000ms, count:  13, total: 21
Total added: 21, Avg rate: 21.00/thrd, 3.50/sec = 285.7msec/op
```

Part 5

LDIF Deployment Tools

Chapter 17

The Directory Server 4.x Instance Creation Tool

The `create_instance.pl` tool is a Perl script that helps automate the process of creating new Sun™ ONE Directory Server 4.*x* instances on a machine where that product has already been installed. This chapter provides instructions on how to use the script. It contains the following sections:

- Overview
- Command Usage
- Configuration Information

Overview

The `create_instance.pl` tool automates the process of creating new Directory Server 4.*x* instances on a machine where that product has already been installed. The script is not compatible with the Sun ONE Directory Server 5.*x* product. The tool will create the new instance using existing configuration files and information given interactively by the user. The Sun ONE Directory Server Resource Kit (DSRK) includes the script in the *DSRK_base*`/perl` directory. It requires Perl version 5.005_03 or later.

NOTE See “Third-Party Sources of Information” on page 32 for links to Perl resources.

Command Usage

The `create_instance.pl` script reads existing configuration files and queries the user for any additional information about the new instance. It creates a file named `newdsinst.inf` that details the configuration for the new instance. (See "Configuration Information" on page 220 for a listing of the collected information.) When finished, the tool will prompt you to create a new instance using the configurations detailed in this file. Following is the procedure to run `create_instance.pl`:

1. Change to the *DirectoryServer_base*`/admin-serv/config` directory.

   ```
   $ cd DirectoryServer_base/admin-serv/config
   ```

2. Run `create_instance.pl` from the command-line.

   ```
   $ ./create_instance.pl
   ```

NOTE The tool has no options, but it must be run from the `config` directory of a previously installed administration server.

Configuration Information

The following sections detail the configuration information written to the `newdsinst.inf` file (as generated by the `create_instance.pl` tool). The placeholders in *italics* show the origin of the specific information, either from an existing file or through user input.

General Information

This information is listed under the `[general]` section of the `newdsinst.inf` file.

- FullMachineName= *value of ldapHost from adm.conf file*
- SuiteSpotUserID= *user input of OS user ID*
- SuitespotGroup= *user input of OS group ID*
- ServerRoot= *value of ServerRoot from local.conf file*
- AdminDomain= *value of isie from adm.conf file*
- ConfigDirectoryAdminID= *administrator ID extracted from admpw file*

- ConfigDirectoryAdminPwd= *user input of existing config DS admin password*
- ConfigDirectoryLdapURL= ldap://*ldapHost*:*ldapPort*/ (from `adm.conf` file)
- Components= svrcore,base,slapd,admin

New Directory Instance Information

This information is listed under the `[slapd]` section of the `newdsinst.inf` file.

- SlapdConfigForMC= No
- SecurityOn= No
- UseExistingMC= Yes
- UseExistingUG= No
- ServerPort= *user input of new server port*
- ServerIdentifier= *user input of new server identifier*
- Suffix= *user input of new data suffix*
- RootDN= *user input of new root DN*
- UseReplication= No
- SetupSupplier= No
- SetupConsumer= No
- AddSampleEntries= No
- InstallLdifFile= none
- AddOrgEntries= No
- DisableSchemaChecking= No
- RootDNPwd= *user input of new root DN password*
- Components= slapd,slapd-client

Administration Information

This information is listed under the `[admin]` section of the `newdsinst.inf` file.

- SysUser= *value of configuration.nssuitespotuser in local.conf file*

- Port= *value of configuration.nsserverport in local.conf file*
- ServerIpAddress= *value of configuration.nsadminaccessaddresses in local.conf file*
- ServerAdminID= *administrator ID extracted from admpw file*
- ServerAdminPwd= *user input of existing config DS admin password*
- Components= admin,admin-client

Base Information

This information is listed under the [base] section of the newdsinst.inf file.

- Components= base,base-client,base-jre

Chapter 18

The Standard Schema LDIF Generator Tool

The `dbgen.pl` tool generates a sample database containing entries with random values. The entries can then be loaded into a directory server and used to run performance tests. This chapter provides instructions on how to use the script. It contains the following sections:

- Overview
- Command Usage
- Sample Output

Overview

The `dbgen.pl` (database generator) tool is a Perl script that generates a database containing entries with random attribute values. The output is an LDAP Data Interchange Format (LDIF) text file which can be loaded into a directory server and used to run performance tests. Generated entries follow the standard schema to provide more realistic output.

NOTE The `ldifgen` tool, detailed in Chapter 19, "The Custom Schema LDIF Generator Tool," generates entries using a custom schema. For this reasons, `dbgen.pl` is the recommended tool for random database creation.

The DSRK includes the tool in the *DSRK_base*`/perl` directory.

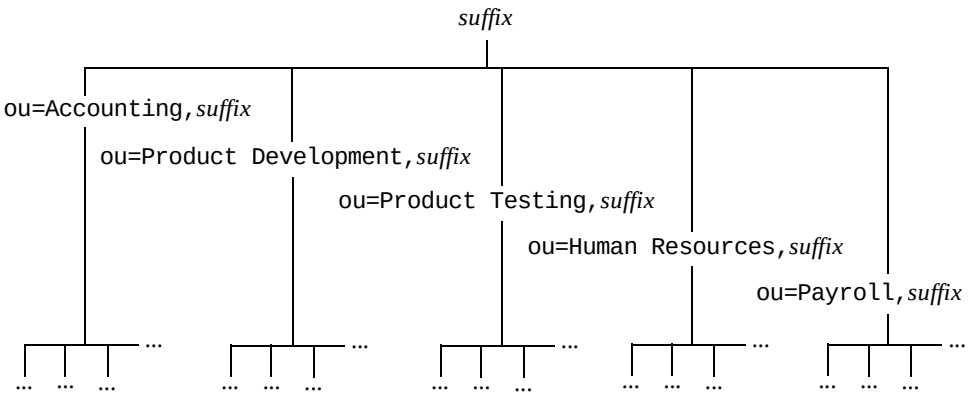
NOTE This script requires Perl version 5.005_03 or later. See "Third-Party Sources of Information" on page 32 for links to Perl resources.

Accompanying Data Files

The script relies on the following files in the *DSRK_base*/data directory:

- dbgen-OrgUnits contains the names for five organizational units (Accounting, Product Development, Product Testing, Human Resources, and Payroll). One entry for each organizational unit (ou) will be created under the suffix root of the generated directory. Then, each person entry will be randomly placed in one of the organizational units. Figure 18-1 shows the directory tree representing the LDIF entries of these organizational units.

Figure 18-1 LDAP Directory Tree Generated by dbgen.pl



- dbgen-GivenNames and dbgen-FamilyNames contain over 8600 plausible first names and 13400 plausible last names, respectively. Each entry representing a person is based on a given name and surname, each randomly chosen from these files. A sequential integer is also appended to all names to ensure that no two entries and no two DNs are identical.

Some attribute values such as email addresses are derived from the name and the suffix to create a plausible entry. Other data such as telephone numbers or titles are generated randomly or selected randomly from internal lists. These types of attribute values are not configurable.

NOTE You may edit the contents of these data files to modify the output of the dbgen.pl tool. (You will need root privileges to edit these files on UNIX®-based systems.)

Customizing the Script

If you customize the `dbgen.pl` script for added functionality, we encourage you to share your work with other LDAP users. Please post a message to the `iplanet.server.idsrk` public newsgroup with your ideas or your code.

Command Usage

The `dbgen.pl` tool generates a directory hierarchy that mimics a simple corporate structure. The script has an option for specifying a *suffix*, which will be the root of the generated hierarchy. (By default it is `dc=siroe,dc=com`..) All of the leaf entries are `inetOrgPerson` objects by default, or you may specify the `-O` (the capital letter O) option to create all `OrganizationalPerson` objects. The tool generates random values for either object class's allowed attributes. (See the "Sample Output" on page 226 for examples of both object classes and the generated attribute values.)

The tool also generates the root and `ou` entries so that its output is a complete and valid LDAP directory hierarchy expressed in LDIF text. The `ou` entries do not contain any attributes and are always the same. The root entry contains only simple `aci` (Access Control Instructions) attributes that are also invariant.

Syntax

The syntax of the `dbgen.pl` tool on the command-line takes the following form:

```
dbgen.pl -o filename.ldif -n number [options]
```

Where:

- *filename*`.ldif` is a writable file that will contain the LDIF output.
- *number* is the number of leaf entries that will be generated, in addition to the parent entries that are always present.

CAUTION Due to the implementation of randomizing functions on certain platforms, the `dbgen.pl` tool may generate exactly the same attribute values every time it is invoked. While the values are seemingly random, they may in fact be identical to the other entries with the same DN created by different invocations of the command. Using the `-r` *seed* option will control this phenomenon. Using a different seed with every invocation, such as a timestamp, will ensure that output is always different. Invocations that use the same seed may produce the same output.

Options

The dbgen.pl options and parameters are described in Table 18-1. Running the dbgen.pl script from the command-line without any options or parameters will display the usage help text that briefly describes all options.

Table 18-1 Command-Line Options for dbgen.pl

Option	Parameter	Purpose
-o	*outputFile*	Specify the output file for the generated database. The output file is a complete directory hierarchy in LDIF text. This parameter is required.
-n	*number*	Specify the number of leaf entries in the generated database. This parameter is required.
-s	*suffix*	Specify the root of the generated hierarchy. The tool will create an entry of class top with a DN equal to the *suffix*, and the DN of all other generated entries will contain this *suffix*. The suffix must begin with either dc=*DomainComponent* or o=*Organization*. When this option is omitted, the default suffix is dc=siroe,dc=com.
-c		Use the cn (common name) attribute and value in the RDN. When this option is ommited, the uid attribte will be used.
-O	(capital letter O)	Generate all leaf entries as OrganizationalPerson objects and create only the corresponding attributes. When this option is omitted, all leaf entries will have the object class inetOrgPerson and the additional attributes of this class.
-r	*seed*	Specify a *seed* integer for the random number generator. The data generated by the tool will be different from one execution to the next only if the seed is different. A common way to ensure the seed is different with every execution is to use a timestamp.
-p		This option is mentioned in the online help, but it is deprecated.
-q		Quiet output mode: dbgen.pl will not display any measure of progress while running. When this flag is omitted, the tool will display a line of dots, one dot for every 10,000 entries generated.
-v		Verbose output mode: dbgen.pl will display additional messages about its progress.

Sample Output

The examples in this section show the output of the dbgen.pl tool. These examples use the default data files found in the *DSRK_base*/data directory. To save space, only the first full leaf entry is shown in each case.

InetOrgPerson Entries

This example demonstrates how the `dbgen.pl` script generates random attribute values for entries of the `inetOrgPerson` object class. It uses the `date` command of the UNIX® shell to generate a seed for the random number generator. It also demonstrates the verbose output that includes progress messages.

Code Example 18-1 InetOrgPerson Generated Entry

```
$ perl dbgen.pl -o out1.ldif -n 3 -r ‘date +%S‘ -v
Loading Name Data...
Done
Ok, now generating 3 entries, please wait
.Generated 3 entries, 0 duplicates skipped
$ cat out1.ldif
dn: dc=siroe,dc=com
objectClass: top
objectClass: domain
dc: siroe
aci: (target=ldap:///dc=siroe,dc=com)(targetattr=*)
 (version 3.0; acl "acl1"; allow(write) userdn = "ldap:///self";)
aci: (target=ldap:///dc=siroe,dc=com)(targetattr=*)
 (version 3.0; acl "acl2"; allow(write) groupdn =
 "ldap:///cn=Directory Administrators, dc=siroe,dc=com";)
aci: (target=ldap:///dc=siroe,dc=com)(targetattr=*)
 (version 3.0; acl "acl3"; allow(read, search, compare) userdn =
 "ldap:///anyone";)
dn: ou=Accounting, dc=siroe,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Accounting
dn: ou=Product Development, dc=siroe,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Product Development
dn: ou=Product Testing, dc=siroe,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Product Testing
dn: ou=Human Resources, dc=siroe,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Human Resources
dn: ou=Payroll, dc=siroe,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Payroll
dn: uid=HDiogo0, ou=Product Testing, dc=siroe,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Harry Diogo
sn: Diogo
```

Code Example 18-1 InetOrgPerson Generated Entry *(Continued)*

```
uid: HDiogo0
givenName: Harry
description: This is Harry Diogo0's description
userPassword: HDiogo0
departmentNumber: 5083
employeeType: Manager
homePhone: +1 804 339-8183
initials: H. D.
telephoneNumber: +1 818 605-3216
facsimileTelephoneNumber: +1 206 768-6857
mobile: +1 408 194-1108
pager: +1 415 692-3053
manager: Pippy Mejdal0
secretary: Eden Yvon0
roomNumber: 2265
carLicense: ORHYA5I
l: Redwood Shores
ou: Product Testing
mail: Harry_Diogo@siroe.com
postalAddress: Product Testing Dept #441, Room#73
title: Senior Product Testing Guru
dn: uid=MPlanta1, ou=Product Development, dc=siroe,dc=com
[...]
dn: uid=GVela2, ou=Human Resources, dc=siroe,dc=com
[...]
```

OrganizationalPerson Entries

In this example we use the `-O` (capital letter O) option to generate entries of the `organizationalPerson` object class. We also use the `-c` and `-s` options to customize the DN of the generated entries.

Code Example 18-2 OrganizationalPerson Generated Entry

```
$ perl dbgen.pl -o out2.ldif -n 3 -O -r 'date +%S' -q \
                -c -s "o=Varrius Corp.,c=US"
$ cat out2.ldif
dn: o=Varrius Corp.,c=US
objectClass: top
objectClass: organization
o: Varrius Corp.
aci: (target=ldap:///o=Varrius Corp.,c=US)(targetattr=*)
 (version 3.0; acl "acl1"; allow(write) userdn = "ldap:///self";)
aci: (target=ldap:///o=Varrius Corp.,c=US)(targetattr=*)
 (version 3.0; acl "acl2"; allow(write) groupdn =
 "ldap:///cn=Directory Administrators, o=Varrius Corp.,c=US";)
aci: (target=ldap:///o=Varrius Corp.,c=US)(targetattr=*)
 (version 3.0; acl "acl3"; allow(read, search, compare) userdn =
```

Code Example 18-2 OrganizationalPerson Generated Entry *(Continued)*

```
 "ldap:///anyone";)
dn: ou=Accounting, o=Varrius Corp.,c=US
objectClass: top
objectClass: organizationalUnit
ou: Accounting
dn: ou=Product Development, o=Varrius Corp.,c=US
objectClass: top
objectClass: organizationalUnit
ou: Product Development
dn: ou=Product Testing, o=Varrius Corp.,c=US
objectClass: top
objectClass: organizationalUnit
ou: Product Testing
dn: ou=Human Resources, o=Varrius Corp.,c=US
objectClass: top
objectClass: organizationalUnit
ou: Human Resources
dn: ou=Payroll, o=Varrius Corp.,c=US
objectClass: top
objectClass: organizationalUnit
ou: Payroll
dn: cn=Mihaela Inman, ou=Accounting, o=Varrius Corp.,c=US
objectClass: top
objectClass: person
objectClass: organizationalPerson
cn: Mihaela Inman
sn: Inman
uid: MInman0
givenName: Mihaela
description: This is Mihaela Inman's description
userPassword: MInman0
telephoneNumber: +1 714 803-8455
facsimileTelephoneNumber: +1 714 371-9310
l: Santa Clara
ou: Accounting
mail: Mihaela_Inman@siroe.com
postalAddress: Accounting Dept #339, Room#98
title: Chief Accounting Director
dn: cn=Randall Braddy,ou=Human Resources,o=Varrius Corp.,c=US
[...]
dn: cn=Morley Cantwell,ou=Product Development,o=Varrius Corp.,c=US
[...]
```

The Custom Schema LDIF Generator Tool

The `ldifgen` tool creates Lightweight Directory Access Protocol (LDAP) entries with randomly generated content. The entries can then be loaded into an LDAP server to create a test directory. This chapter provides instructions on how to use the command. It contains the following sections:

- Overview
- Command Usage
- Sample Output

Overview

The `ldifgen` (LDIF generator) tool creates LDAP entries with randomly generated content. The output is generated as LDIF (LDAP Data Interchange Format) text that can be loaded into an LDAP server to create a test directory. As with the `dbgen.pl` script (detailed in Chapter 18, "The Standard Schema LDIF Generator Tool"), this generated file can be used to run performance tests.

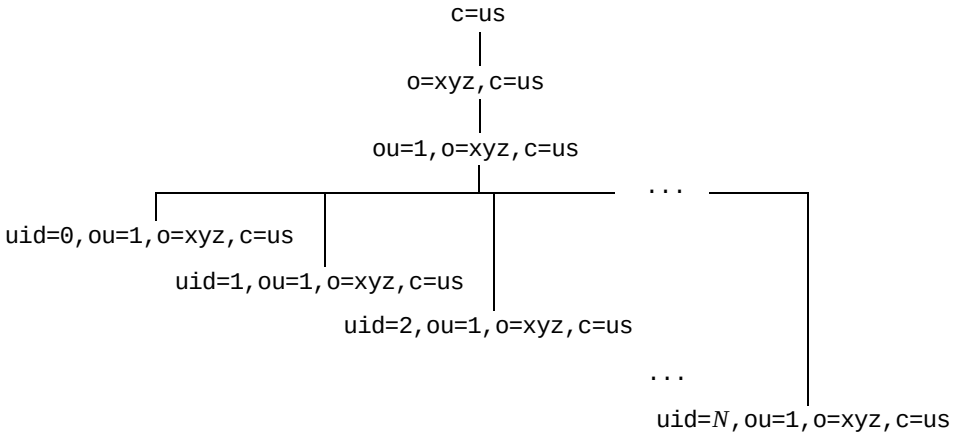
NOTE The difference between the two tools is that `ldifgen` uses a custom schema, creating a simple directory structure with leaf entries belonging to the `xyzmember` object class; `dbgen` uses a standard schema.

After generating the LDIF file, you may use the `ldapmodify` tool to import both the schema and the generated entries. (See Chapter 4, "The ldapmodify Tool" for information.)

NOTE The xyzmember schema defines this custom object class and is generated by one of the command-line options. See “Options” on page 233 for more information.

The DNs of generated entries fit within the minimal hierarchy shown in Figure 19-1 on page 232. The structure of this tree is not configurable although the number of leaf entries is determined by the `-n` option on the command-line.

Figure 19-1 LDAP Directory Hierarchy Created by `ldifgen`



The `ldifgen` tool generates the parent entries so that its output is a complete and valid LDAP directory hierarchy expressed in LDIF text. The three parent entries do not contain any attributes and are always the same.

CAUTION The DNs of generated entries use sequential numbering to ensure that no two entries of the same invocation will be identical. However, due to the implementation of randomizing functions on certain platforms, the `ldifgen` tool may generate exactly the same attribute values every time it is invoked. While the values are seemingly random, they may in fact be identical to the other entries with the same DN created by different invocations of the command.

Command Usage

The `ldifgen` tool generates entries that are `xyzmember` objects. This custom object class acts as a template, and the tool generates random values for each of its attributes. If you do not already have the schema for this object class, use the `-S` option to generate the version corresponding to your directory. You will need to load this schema entry into the directory before loading the generated entries. (An example of this class and its attributes is shown in "Sample Output" on page 234.)

Syntax

The syntax of the `ldifgen` tool on the command-line can take one of two forms:

```
ldifgen -n number > filename.ldif
```

```
ldifgen -S version > filename.ldif
```

Where:

- *number* is the number of leaf entries that will be generated, in addition to the 3 hierarchical entries that are always present.
- *version* is the major version number, either 4 or 5, of the Sun™ ONE Directory Server instance into which the generated entries will be loaded. Using this option, the `ldifgen` tool will output the corresponding schema for the `xyzmember` class and its attributes. The schema is given as an LDIF update statement that may be loaded using the `ldapmodify` command. (See Chapter 4, "The ldapmodify Tool" for information.)
- *filename*`.ldif` is a writable file that will contain the LDIF output.

NOTE The `ldifgen` tool uses the standard output, so you need to redirect this to a file in order to save it in the LDIF.

Options

The `ldifgen` options and parameters are described in Table 19-1. Running the `ldifgen` command without any options or parameters will display the brief usage help text.

Table 19-1 Command-Line Options for `ldifgen`

Option	Parameter	Purpose
`-s`		Generates the version of the custom schema corresponding to your directory. The schema is given as an LDIF update statement that may be loaded using the `ldapmodify` command. See Chapter 4, "The ldapmodify Tool" for information.
`-n`		Specifies the number of leaf entries.
none		Display the usage help text that briefly describes all options.

Sample Output

Code Example 19-1 shows a sample output of the `ldifgen` tool. Only the first leaf entry is shown in its entirety. Some attribute values have been shortened to fit on a single line.

Code Example 19-1 Sample Output From ldifgen

```
$ ldifgen -n 3
dn: C=US
c: US
objectclass: top
objectclass: country
dn: O=XYZ,C=US
o: XYZ
objectclass: top
objectclass: organization
dn: OU=1,O=XYZ,C=US
ou: 1
objectclass: top
objectclass: organizationunit
dn: UID=0,OU=1,O=XYZ,C=US
objectclass: top
objectclass: xyzmember
uid: 0
sponsorId: 18080291083208191342756198584726724745154694425751126868
cn: fozepdkdnqlhfejdzksc cfykdxnlorwvfwavbmyq
login: cl1ldo51y5fam665jwlxvhpxdnyuj6z2y6dhl88p
userpassword: 00000000000000000000000000000000000000000000000000000000
postalCode: 0080227
st: jq
cityCode: iwk
city: 49i0zztw1dsrvw3vvcb8siyhghio0aea921ys00z
cityIds: 3yw4lqpm870qv4d1clwi10ig9pk7gaw9zykquf2sc99x3c5l4rtvx22whs
l: tb3y0tuky2
signId: 333823636224004081689176513341474779054987309927207145798 05
createTimestamp: 20000101000001Z
```

Code Example 19-1 Sample Output From ldifgen *(Continued)*

```
modifyTimestamp: 20000101000001Z
mail: cl1ldo51y5fam665jwlxvhpxdnyuj6z2y6dhl88p@xyz.com
topics: gb1cuwjzi0083xogiyleo3kae3w841xaios2r27zsddlg77ovit9rjqvssu
searchFlag: jx2rblz8tddbgg2ndzhy7gep5aqosa0hdtefkjt3mxk93cmn19regb9
myPageTitle: 6wwavgv5qeei1ohbq9ox9l1fn1wohc4tyqtzdeutjy1qcpr8olvnx2
customization: xy01vjqy4qxde49wjrayivb5nalnvg9ykzplplpj1s2ln1nwyn0n
defBookmarks: wa7mz2eudtjkcbb5jornhi0bq5ie5jn7zkj4c3pvny55g6050xi7l
hints: ly8qoxl6r0rk42faell48ahkpvrfy3gp7u6mhxiyngoxotet4p3jpr9ksq5w
birthdate: zbcxbo9jbxn4j27y1bxc76lmnkx6g886qatr48gl
flags: buvl2tptucvr8mkmcnyjlwf1dyjnc5zdpkoh33h4li1kvbcx4gmgirn1vcfx
sourceAppId: 8khnubb4lyewm5m7n4bha3nzlhq6jbahlhbmysw9w0w8vom8j3wzwp
timeZone: 62
c: fo
pageRefresh: 3647
layout: ocra2dp3r1igak6wzkq2bgeo6d8zuog86xpegknebbkrlgtb7938f72rq0g
colors: u356qkqdu4hb5r8heryrrr5w4yu040h59ch62y92sgj3dhqxyeefbaile2c
givenName: fozepdkdnqlhfejdzksc
sn: cfykdxnlorwvfwavbmyq
address: 8jz817ktvj1t5a2ohjc53wrlczwg7evl30zux9tyykhtkqo7f638s0zv5m
phone: 63714139647-45991442
occupation: opsxdliwvjmemuapxydvewsddzwznyfcozztmjsi
householdIncome: 2620083169
gender: F
friend: uid=0,ou=1,o=xyz
dn: UID=1,OU=1,O=XYZ,C=US
objectclass: top
objectclass: xyzmember
uid: 1
...

dn: UID=2,OU=1,O=XYZ,C=US
objectclass: top
objectclass: xyzmember
uid: 2
...
```

Chapter 20

The Java-based LDIF Generator Utility

MakeLDIF is a Java™-based utility that generates LDAP Data Interchange Format (LDIF) files for importation into a Lightweight Directory Access Protocol (LDAP) directory server. This chapter provides instructions on how to use MakeLDIF. It contains the following sections:

- Overview
- Command Usage
- Customizing the Template File
- Additional Information

Overview

MakeLDIF is a Java-based utility for generating LDIF files. Because the program is written in Java, it requires a Java runtime environment to function properly. The MakeLDIF program was designed to work with Java 1.2 and higher, although more recent versions typically exhibit better performance. The information needed to generate the entries is specified in a template file which can be customized to produce LDIF files for a wide variety of scenarios. The tool can be found in the *DSRK_base*/java/MakeLDIF directory.

NOTE This chapter provides basic information required to use MakeLDIF. See the *MakeLDIF Usage Guide* for more specific information about MakeLDIF features and capabilities. It is located in the *DSRK_base*/java/MakeLDIF directory.

Command Usage

MakeLDIF is used to generate sample data for importation into an LDAP directory server. Other utilities do exist for this purpose but MakeLDIF offers features not available in those tools. The command-line options can be used to control these features and other aspects of the input and output.

NOTE In order to use MakeLDIF, you should be in the same directory as the MakeLDIF.jar file. By default, the directory is *DSRK_base*/java/MakeLDIF.

Syntax

The syntax of the MakeLDIF program on the command-line can take either of the following forms:

```
java -jar MakeLDIF.jar -t example.template -o output.ldif
java -cp MakeLDIF.jar MakeLDIF -t example.template -o output.ldif
```

This syntax will use the template defined in the file *example*.template and write the output into a file titled *output*.ldif.

NOTE If the Java executable is not currently in your path, then you may specify the absolute path to the Java executable you wish to use.

Options

The MakeLDIF program has options available to customize the way that it operates. These are detailed in Table 20-1. The MakeLDIF -H command and option when run on the command-line will display brief descriptions of the command syntax, options, and parameters.

Table 20-1 Options for MakeLDIF

Option	Parameter	Purpose
-f	*filename*	Specifies the name of the file containing first names to use in the entry generation process. By default, the MakeLDIF program will use a file named first.names included in the current working directory. If a custom first name file is used, it should contain only first name information with one first name per line.

Table 20-1 Options for MakeLDIF *(Continued)*

Option	Parameter	Purpose
`-l`	*filename*	Specifies the name of the file containing last names to use in the entry generation process. By default, the `MakeLDIF` program will use a file named `last.names` included in the current working directory. If a custom last name file is used, it should contain only last name information with one last name per line.
`-t`	*filename*	Specifies the template file that will be used to determine how to create the LDIF file. This is a required parameter. The format of the template file is discussed in Customizing the Template File.
`-o`	*filename*	Specifies the name of the output LDIF file to generate. This is a required parameter.
`-d`	*filename*	Specifies the name of a file to which the DNs of generated entries will be written. This is useful for a number of utilities that can make use of a DN file.
`-b`	*filename*	Specifies the name of a file to which bind information for the generated entries will be written. The format will be one user per line with the DN followed by a tab and the password for the user with that DN.
`-L`	*filename*	Specifies the name of a file to which login information for the generated entries will be written. The format will be one user per line with the login ID followed by a tab and the password for that user.
`-i`	*attribute*	Specifies the name of the attribute that should be used as the login ID. By default, the uid attribute will be used as the login ID.
`-F`	*filename*	Specifies that search filters constructed from the generated entries will be written to the specified file. You will need to use the "-T" option to specify which filter types to generate.
`-T`	*type*	Specifies filter types that should be generated. The format of *type* is the name of the attribute followed by a colon and a comma-separated list of filter types to create for that attribute. Allowable filter types are "eq" and "sub". Example: "-T uid:eq -T cn:eq,sub".
`-s`	*value*	Specifies the seed to use for the random number generator. If no seed is specified, then a value will be chosen based on the current time. If a positive integer value is given, then that will be used as the seed to the random number generator. Using the same template file and the same random seed should consistently produce exactly the same LDIF file.
`-m`	*value*	Specifies the maximum number of entries that should be written to a single LDIF file. After this number of entries has been written, the current LDIF file will be closed and a new file created with a ".2" extension. This counter will increment sequentially for each new file that is created.

Table 20-1 Options for MakeLDIF *(Continued)*

Option	Parameter	Purpose
`-x`	*value*	Specifies the maximum number of entries that should be created for each template below each branch. This can be used to create an example of the LDIF file that would be generated from the provided template to verify that the template syntax is valid.
`-w`		Specifies that long lines should be wrapped (folded). By default, the entire value of an attribute is written on a single line, which is useful if the LDIF file needs to be processed by another application, but can be difficult to read for long values.
`-M`		Specifies that a different filter file should be created for each index type for each attribute for which filter information is to be collected.
`-S`		Specifies that branch entries should be skipped when writing the LDIF information to the requested output file.
`-H`		Displays usage information for the `MakeLDIF` program.
`-V`		Displays version information for the MakeLDIF program.

Customizing the Template File

The `MakeLDIF` program uses a template file to customize the LDIF file that is generated. The use of the template file makes it possible to customize the location and kinds of entries that are generated by `MakeLDIF`. There are several kinds of entries that may be written into a template file: Global Replacement Definitions, Branch Entry Definitions and Template Definitions. `example.template` is provided with `MakeLDIF` in the *DSRK_base*`/java/MakeLDIF` directory. Definitions from this file are used in the following sections as examples.

Global Replacement Definitions

The first section in a template file usually defines global replacement variables. These variables are used for reference in the template file itself. Placeholders will be automatically replaced by the defined variable as each line is read into memory. Code Example 20-1 are the global replacement definitions from `example.template`.

Code Example 20-1 Global Replacement Entries From example.template

```
define suffix=dc=example,dc=com
define maildomain=example.com
define numusers=10000
```

As an example, the first entry (`define suffix=dc=example,dc=com`) will create a global replacement variable named *suffix* with a value of `dc=example,dc=com`. Now, any case in which *suffix* appears in square brackets (e.g., "[suffix]") will be replaced with the value that has been defined for it. Global replacement definitions must appear at the top of the file. However, it is not required that there be any; if there are none, the template file would start with the Branch Entry Definitions.

Branch Entry Definitions

Branch entry definitions define the basic structure for the directory server. They specify the entry (or entries) that should appear at the top of the hierarchy as well as the number and kinds of entries that should appear below those parent entries. Code Example 20-2 are the branch entry definitions from `example.template`.

Code Example 20-2 Branch Entry Definitions From example.template

```
branch: [suffix]
aci: (targetattr!="userPassword")(version 3.0; acl "Anonymous access"; allow
(read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr != "nsroledn || aci || nsLookThroughLimit || nsSizeLimit ||
nsTimeLimit || nsIdleTimeout || passwordPolicySubentry ")(version 3.0; acl
"Allow self entry modification except for nsroledn, aci, resource limit
attributes, and passwordPolicySubentry"; allow (write)userdn
="ldap:///self";)
aci: (targetattr = "*")(version 3.0; acl "Configuration Administrator";
allow (all) userdn = "ldap:///uid=admin, ou=Administrators,
ou=TopologyManagement, o=NetscapeRoot";) aci: (targetattr ="*")(version
3.0;acl "Configuration Administrators Group";allow (all) (groupdn =
"ldap:///cn=Configuration Administrators, ou=Groups, ou=TopologyManagement,
o=NetscapeRoot");)
aci: (targetattr ="*")(version 3.0;acl "Directory Administrators
Group";allow (all) (groupdn = "ldap:///ou=Directory Administrators,
[suffix]");)

branch: ou=People,[suffix]
subordinateTemplate: person:[numusers]
```

Code Example 20-2 Branch Entry Definitions From example.template *(Continued)*

```
aci: (targetattr ="userpassword || telephonenumber ||
facsimiletelephonenumber")(version 3.0;acl "Allow self entry
modification";allow (write)(userdn = "ldap:///self");)
aci: (targetattr !="cn || sn || uid")(targetfilter
="(ou=Accounting)")(version 3.0;acl "Accounting Managers Group
Permissions";allow (write)(groupdn = "ldap:///cn=Accounting
Managers,ou=groups,[suffix]");)
aci: (targetattr !="cn || sn || uid")(targetfilter ="(ou=Human
Resources)")(version 3.0;acl "HR Group Permissions";allow (write)(groupdn =
"ldap:///cn=HR Managers,ou=groups,[suffix]");)
aci: (targetattr !="cn ||sn || uid")(targetfilter ="(ou=Product
Testing)")(version 3.0;acl "QA Group Permissions";allow (write)(groupdn =
"ldap:///cn=QA Managers,ou=groups,[suffix]");)
aci: (targetattr !="cn || sn || uid")(targetfilter ="(ou=Product
Development)")(version 3.0;acl "Engineering Group Permissions";allow
(write)(groupdn = "ldap:///cn=PD Managers,ou=groups,[suffix]");)
```

Basic Structure of Branch Entry Definition

The basic structure of the branch entry definition is defined by the RDN attribute specified in the DN of the branch definition. In Code Example 20-2, the initial branch defintion is defined as:

```
branch: [suffix]
```

or:

```
branch: dc=example,dc=com
```

`MakeLDIF` associates the RDN with specific branch entries. The default entries created based on RDN are:

- `dc` -- Creates an entry with the `domain` objectclass.
- `o` -- Creates an entry with the `organization` objectclass.
- `ou` -- Creates an entry with the `organizationalUnit` objectclass.
- `c` -- Creates an entry with the `country` objectclass.
- `l` -- Creates an entry with the `locality` objectclass.

NOTE It is possible to use other kinds of RDN attributes for a branch entry than those listed. For branch entries with other RDN attributes, the entry will be created with the `extensibleObject` objectclass.

Thus, the initial branch definition specifies that an entry with the `domain` objectclass should be created based on the *suffix* global replacement variable detailed in Code Example 20-1. This means that an entry `dc=example,dc=com` should be created. In LDIF, the entry will look like Code Example 20-3.

Code Example 20-3 LDIF of Basic Branch Definition

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

The branch defined entry will contain only the DN, its two objectclass values, and the RDN attribute. It is possible to include other attributes in the branch entry by including them in the branch definition of the template file. Code Example 20-4 adds a description attribute to the basic branch definition.

Code Example 20-4 Extended Branch Definition Entry

```
branch: dc=example,dc=com
description: This is the description.
```

Code Example 20-4 will create an entry that looks like Code Example 20-5.

Code Example 20-5 LDIF of Extended Branch Definition Entry

```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example
description: This is the description.
```

This additional text is static and, unlike template definitions that may occur later in the template file, branch definitions are not dynamically parsed.

SubordinateTemplate Definition Entries

The branch entry defined thus far created only a single entry in the directory with limited control over the information contained in that entry. However, if one or more `subordinateTemplate` values are specified, it is possible to create additonal entries below the initial branch. For example, the first branch entry definition detailed in Code Example 20-2 will not cause any entries to be created below it. On the other hand, the second definition (further on in Code Example 20-2) will create subordinate entries because a `subordinateTemplate` is defined:

```
branch: ou=People,[suffix]

subordinateTemplate: person:[numusers]
```

NOTE Note that `suffix` and `numusers` refer to global replacement entries also defined in the template. See Code Example 20-1.

This `subordinateTemplate` code will create an `ou=People,dc=example,dc=com` entry and 10000 sub-entries created below it, modeled after the person template entry defined later in the file. (See “Template Definitions” on page 245 for more information.)

Branch definition entries are not limited to just one `subordinateTemplate` definition. Multiple `subordinateTemplate` definitions can be defined by including each on separate lines of the branch definition. 1000 entries based on the `person` template and an additional 100 entries based on the `certificatePerson` template will be created by this branch entry definition:

```
branch: ou=People,dc=example,dc=com

subordinateTemplate: person:1000

subordinateTemplate: certificatePerson: 100
```

Global Replacement Variables In Branch Definition Entries

It is possible to use global replacement variables in branch entry definitions. Code Example 20-6 defines both global replacement variables and branch entry definitions.

Code Example 20-6 Global Replacement Variables With Branch Entry Definitions

```
define suffix=dc=example,dc=com
branch: [suffix]
branch: ou=People,[suffix]
subordinateTemplate: 1000
```

Code Example 20-7 details the `domain` and the `organizationlUnit` that are created by Code Example 20-6. 1000 additional entries below them will also be created based on the `person` template.

Code Example 20-7 LDIF of Branch Definition Entries With Global Variables

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

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

Template Definitions

Branch Entry Definitions are used to specify a few key entries in the directory server hierarchy, but template definitions contain a much more useful set of information. They contain information that may be used to generate large numbers of entries below branch entries. Template definitions are parsed and may contain special tags to customize the kinds of entries that are built, so even though the same template may be used to generate thousands or millions of entries, each of those entries may still be unique. Code Example 20-8 are the template definitions from `example.template` provided with `MakeLDIF` in the *DSRK_base*`/java/MakeLDIF` directory.

Code Example 20-8 Template Definitions From example.template

```
template: person
rdnAttr: uid
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
givenName: <first>
sn: <last>
cn: {givenName} {sn}
initials: {givenName:1}{sn:1}
uid: {givenName}.{sn}
mail: {uid}@[maildomain]
userPassword: <random:alphanumeric:8>
telephoneNumber: <random:telephone>
homePhone: <random:telephone>
pager: <random:telephone>
```

Code Example 20-8 Template Definitions From example.template *(Continued)*

```
mobile: <random:telephone>
employeeNumber: <sequential:100000>
street: <random:numeric:5> <file:streets> Street
l: <file:cities>
st: <file:states>
postalCode: <random:numeric:5>
postalAddress: {cn}${street}${l}, {st}  {postalCode}
description: This is the description for {cn}.

template: certificatePerson
rdnAttr: uid
extends: person
userCertificate:: <random:base64:975>

template: messaging52person
rdnAttr: uid
extends: person
objectClass: inetUser
objectClass: inetMailUser
objectClass: inetLocalMailRecipient
objectClass: userPresenceProfile
mailAlternateAddress: {givenName}_{sn}@[maildomain]
mailDeliveryOption: mailbox
mailHost: mail.example.com
mailQuota: -1
mailMsgQuota: -1
inetUserStatus: active
mailUserStatus: active

template: ADperson
rdnAttr: cn
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
givenName: <first>
sn: <last>
cn: {givenName} {sn}
name: {cn}
displayName: {cn}
initials: {givenName:1}{sn:1}
sAMAccountName: {givenName:1}{sn}
countryCode: 0
userPrincipalName: {sAMAccountName}@[maildomain]
unicodePwd:: <base64:UnicodeLittleUnmarked:"<random:alphanumeric:8>">
telephoneNumber: <random:telephone>
userAccountControl: 512
```

The tags contained in these definitions will be parsed and replaced with information appropriate to the specified tag. Code Example 20-9 details an entry made from the first template definition in Code Example 20-8.

Code Example 20-9 LDIF Entry for Template Definition

```
dn: uid=John.Doe,ou=People,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
givenName: John
sn: Doe
cn: John Doe
initals: JD
uid: John.Doe
mail: John.Doe@example.com
userPassword: d82fk32n
telephoneNumber: 823-630-8157
homePhone: 823-766-8231
pager: 823-766-8790
mobile: 823-766-9731
employeeNumber: 124509
street: 12345 Forsaken Street
l: San Jose
st: California
postalCode: 95127
postalAddress: John Doe 12345 Forsaken Street San Jose, California 95127
description: This is the description for John Doe.
```

The first line of a template definition should contain the name of the template. This name identifies the template so that the appropriate entry may be created under branch entries. Each template entry should have a unique name. The template definition should also have an `rdnAttr` line that specifies the attribute to be used for the RDN component of the entry's DN. This must be a single value -- multivalued RDNs are not supported by this version of the LDIF generator. If an `rdnAttr` definition is not present, a default RDN attribute of `cn` will be used.

Tokens Supported in Template Definitions

The information in a template definition (besides the template name and RDN attriubte) will be written into the entries that are generated; any recognized tokens will be evaluated and replaced with the appropriate information. The tokens detailed in Table 20-2 are supported.

Table 20-2 Supported Tokens in Template Definitions

Token	Purpose
`presence:`*percent*	Indicates how likely the associated attribute is to be included in any given entry generated from this template. The given percentage value should be a number between 0 and 100. This should only be used with attributes that are not required by the objectClasses used in the entry, and there should be something else included in the value of that attribute that will be present in entries that are chosen to include the attribute.
first	Replaced with a first name from the first name file. If both a first and last name are included in an entry, the combination of the two is guaranteed to be unique. That is, no two entries in the LDIF file will have the same combination of first and last name values. Note that in order to guarantee uniqueness, the first and last names must be used in their entirety; that is, you cannot use the substring feature of attribute value replacements of the form givenName:5 discussed below.
last	Replaced with a last name from the last name file. If both a first and last name are included in an entry, the combination of the two is guaranteed to be unique. That is, no two entries in the LDIF file will have the same combination of first and last name values. Note that in order to guarantee uniqueness, the first and last names must be used in their entirety; that is, you cannot use the substring feature of attribute value replacements of the form givenName:5 discussed below.
parentdn	Replaced with the DN of the parent entry.
`random:alpha:`*length*	Replaced with a string of randomly-chosen alphabetic characters the length of which is defined by the *length* token.
`random:numeric:`*length*	Replaced with a string of randomly-chosen numeric digits the length of which is defined by the *length* token.
`random:alphanumeric:`*length*	Replaced with a string of randomly-chosen alphanumeric characters the length of which is defined by the *length* token.
`random:hex:`*length*	Replaced with a string of randomly-chosen hexadecimal digits the length of which is defined by the *length* token.
`random:base64:`*length*	Replaced with a string of randomly-chosen base64 characters the length of which is defined by the *length* token. If the specified length is not a multiple of 4, the base64 value produced will be padded with equal signs so that the total length is a multiple of 4.

Table 20-2 Supported Tokens in Template Definitions *(Continued)*

Token	Purpose
random:telephone	Replaced with a string of randomly-chosen numeric digits in the form 123-456-7890. This uses a US-style telephone number, but it is possible to generate telephone numbers in the format used by other countries by combining other random tags. For example, to generate a telephone number in the UK format, you could use: `+44 random:numeric:4 random:numeric:6`.
<sequential> --	Replaced with a sequentially-increasing numeric value. The first value will be zero. Sequential counters are separate on a per-attribute basis, so it is possible to use multiple sequential counters in the different attributes of the same entry without impacting each other.
guid	Replaced with a GUID value containing hexadecimal digits in the form 12345678-90ab-cdef-1234-567890abcdef. GUID values should be unique within the same LDIF file.
`list:`*value1*`,`*value2*`,...,`*valueN*	Replaced with a randomly-chosen value from the comma-delimited list. Each value will have an equal chance of being chosen.
`list:`*value1*`:`*weight1*`,`*value2*`:`*weight2*`,...,`*valueN*`:`*weightN*	Replaced with a randomly-chosen value from the comma-delimited list. The weight associated with each list item determines how likely that value is to be chosen. For example, a list item with a weight of 2 is twice as likely to be chosen as an item with a weight of 1. The weights specified must be positive integers.
`file:`*filename*	Replaced with a randomly-chosen value from the specified file. There should be one value per line of the file. It is not possible to assign weights to the values in a file, but a value can be weighted artificially by making it appear multiple times in the file.
`exec:`*command*	Replaced with the information sent to standard output when the specified command is executed on the system. Note that because this requires a separate process to be invoked for each entry created using this template, using the exec tag can make the LDIF generation process proceed much more slowly than if the exec tag is not used.
`exec:`*command*`,`*arg1*`,`*arg2*`,...,`*argN*	Replaced with the information sent to standard output when the specified command is executed on the system with the given set of arguments. Note that because this requires a separate process to be invoked for each entry created using this template, using the exec tag can make the LDIF generation process proceed much more slowly than if the exec tag is not used.

In addition to the tokens listed above, it is also possible to include the value of another attribute. For example, the string `givenName.sn` will be replaced with the value for the `givenName` attribute and the value for the `sn` attribute separated by a period. Note that if this feature is used, the attribute must be defined in the template before its value is referenced. If the specified attribute has multiple values, then the first value encountered for that attribute will be used. If the form `givenName:5` is used, then only the first five characters of the `givenName` attribute will be used in the replacement. If there are fewer than the specified number of characters in the value of the target attribute, then the entire value of that attribute will be used.

Subordinate Templates in Template Definitions

It is possible to include one or more `subordinateTemplate` definitions (as discussed in "SubordinateTemplate Definition Entries" on page 244) in a template definition. This can be used to create entries generated from one template below entries generated from another template. Although not a recommended practice when designing a directory, it is an approach that some administrators have taken. Note that a template must not define itself as a subtemplate nor can there be any circular references. For example, an entry generated by one template cannot have another entry generated by the same template anywhere beneath that entry in the directory server hierarchy. If this is done, then the LDIF generation process will become stuck in an infinite loop and will eventually fail. Therefore, this feature should be used with caution.

Inheritance in Template Definitions

Template definitions also support the concept of inheritance. Therefore, it is possible to specify a template definition that builds upon a previously-defined template by using the `extends` definition. For example, if there are to be ten thousand entries created using the `person` template, and an additional one thousand entries created that should have the same structure as the `person` entries but should also include a value for the `userCertificate` attribute, you could create a second template entry as detailed in Code Example 20-10.

Code Example 20-10 extends Definition for Inheritance

```
template: certificatePerson
rdnAttr: uid
extends: person
userCertificate: <random:base64:1000>
```

One benefit to using the extends keyword rather than redefining the entire template is that the version using extends will be smaller. Additionally, it will not be necessary to keep changes synchronized between related template entries. A change made in the initial template will automatically be reflected in any templates that extend it. Code Example 20-10 has the same effect as the template definition detailed in Code Example 20-11 without the use of the extends definition.

Code Example 20-11 Template Definition Without extends Definition

```
template: certificatePerson
rdnAttr: uid
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
givenName: <first>
sn: <last>
cn: {givenName} {sn}
uid: {givenName}.{sn}
mail: {uid}@example.com
userPassword: <random:alphanumeric:8>
telephoneNumber: <random:telephone>
userCertificate: <random:base64:1000>
```

It is possible to use multiple levels of inheritance. But as with the subordinateTemplate definition, you must not define a template as its own parent or define any circular references in which a template may in some way extend itself. Also note that any template that extends another template must specify its own RDN attribute. The RDN attribute of the parent template will not automatically be used. This makes it possible to specify a different RDN attribute for one template than the one that is used for the entries created with the parent template.

Additional Information

More information on MakeLDIF can be found in the *MakeLDIF LDIF Generator Usage Guide*, a Portable Document Format (PDF) file located in the *DSRK_base*/java/MakeLDIF directory.

Chapter 21

The LDIF Transformation Tool

The `ldifxform` tool reformats the contents of an LDIF (LDAP Data Interchange Format) text file. This chapter provides instructions on how to use the `ldifxform` tool. It contains the following sections:

- Overview
- Command Usage
- Reformatting Operations
- Command-Line Examples

Overview

The `ldifxform` (LDIF transform) tool reformats the contents of an LDIF text file. This tool can perform many transformations on LDIF input, such as converting between all of the most common character sets, extracting attribute values, modifying attribute names, ordering entries based on attribute values, or giving detailed statistics. In all cases, modifications are written to a new LDIF file, and the input file is never changed.

TIP `ldifxform` can also be used to analyze or edit the contents of a directory offline by processing the LDIF output of the `db2ldif` tool. For more information, see "db2ldif (Export Database Contents to LDIF)" in Chapter 2 of the *Sun ONE Directory Server Reference Manual*).

Command Usage

The `ldifxform` tool acts as a stream filter, reading input from one file, performing transformations and writing the output to another file. Each transformation is specified by a *command* parameter on the command-line. Several compatible transformations may be performed simultaneously.

All possible operations are listed in "Reformatting Operations" on page 255. Some produce LDIF output destined to be reloaded into a directory. For example, renaming an attribute can be more easily processed on an LDIF file than online through requests to a directory server. Other reformatting operations do not produce LDIF; they are intended to provide an analysis of directory contents. For example, you may extract all different values of a specific attribute and list them under the DN in which they occur. The statistical operations provide counts of entries and attributes.

Syntax

The syntax of the `ldifxform` tool on the command-line takes the following form:

```
ldifxform [-i input.ldif] [-o outputFile] -c "command" ...
```

Where:

- *input*`.ldif` is a readable file that contains the LDIF text input.
- *outputFile* is a writable file that will contain the reformatted LDIF. Some transformations and character conversions will produce output that is not valid LDIF.
- *command* is one of the supported operations to perform on the input, usually enclosed in double quotes ("") for the shell. See "Reformatting Operations" on page 255 for the list of all available commands. You may specify multiple commands, each preceded by `-c`, if they are mutually compatible.

NOTE The input and output parameters are optional; the tool will use the standard input and output if either or both are omitted. However, the use of files is recommended for portability as standard 8-bit input and output are not fully supported on the Windows® platform.

Options

The `ldifxform` options and parameters are described in Table 21-1. The `ldifxform -h` command will display usage help text that briefly describes all options.

Table 21-1 Command-Line Options for the `ldifxform` Tool

Option	Parameter	Purpose
`-i`	*input*.ldif	Specify the input file that contains the LDIF text to process. When this option is omitted, the tool will read the standard input.
`-o`	*outputFile*	Specify the output file for the reformatted LDIF result. Note that some operations do not produce LDIF output. When this option is omitted, the tool will write to the standard output.
`-c`	*command*	Specify an operation for the tool to apply to the input. The *command* parameter is one of the transformations described in "Reformatting Operations" on page 255. This option may be repeated on the command-line when the corresponding operations are compatible.
`-h`		Display the usage help text that briefly describes all options.

Reformatting Operations

The following sections list the transformations available through the `-c` *command* parameter of the `ldifxform` tool. The commands are grouped by type:

- LDIF Transformations
- Character Set Conversions
- Attribute Modifications
- LDAP Entry Presorting
- Directory Analysis

LDIF Transformations

The LDIF transformations affect the encoding and general appearance of LDIF text files.

Table 21-2 LDIF Text Transformations Using `ldifxform`

Command	Formatting Effect
`-c nob64`	Will undo any base-64 transformations. Note that the output will not be reparsable if there are any binary-valued attributes or attributes beginning with special characters.
`-c sevenbit`	Reformats the output as seven-bit characters by base-64 encoding any attribute values that contain non-ASCII bytes. The output is always reparsable.
`-c longlines`	Prevents `ldifxform` from wrapping lines at the 79th column in the output file. This argument is necessary when editing an LDIF file using a UTF-8 aware editor as a character may be wrapped in the middle of its encoding. The output will be reparsable, but many popular system tools (such as `grep` or `sed`) may not be able to handle lines longer than 1024 characters.
`-c notypes`	Removes attribute type names, therefore the output is no longer LDIF. This is useful for generating a list of values. (See "Command-Line Examples" on page 259.)
`-c nodn`	Removes distinguished names, therefore the output is no longer LDIF. This is also useful for generating a list of values.
`-c cleanzero`	Removes trailing zero bytes from attribute values. This option is needed only when processing an LDIF file from a buggy encoder.

Character Set Conversions

The `ldifxform` tool can be used to convert LDIF files to different *charsets* (character sets). Conversions are useful for porting LDIF files between platforms and for use in directories that require localized data. When porting between platforms, you must ensure that all data in the original can be represented in the target charset.

Table 21-3 Character Set Conversions Using `ldifxform`

Command	Formatting Effect
`-c from=88591`	Converts the input file from the ISO-8859-1 charset into the UTF-8 charset. This conversion allows the source data to be written using ISO-8859-1 text editors.
`-c to=88591`	Converts the input file from UTF-8 into the ISO-8859-1 charset. The output is no longer LDIF, and characters that cannot be represented in ISO-8859-1 will be stripped out.

Table 21-3 Character Set Conversions Using `ldifxform` *(Continued)*

Command	Formatting Effect
`-c from=t61`	Converts the input file from the T.61 charset into the UTF-8 charset. This option should be used when converting data obtained from an X.500 or LDAPv2 servers that used T.61 charset encoding by default.
`-c to=t61`	Converts the input file from UTF-8 into the T.61 charset. The output is no longer LDIF, and characters that cannot be represented in T.61 will be stripped out.
`-c from=mstext` `-c to=mstext`	This pair of commands converts between UTF-8 and the Windows Unicode Text file format.
`-c from=`*charSet* `-c to=`*charSet*	Additional transformations are supported between UTF-8 and the following platform-specific *charSet*s. Platform-specific transformations will result in data loss for values that cannot be represented in the target charset: **Solaris platform:** 646, 8859-1, 8859-2, 8859-3, 8859-4, 8859-5, 8859-6, 8859-7, 8859-8, 8859-9, 8859-10, eucJP, gb2312, iso2022, KOI8-R, PCK, SJIS, UTF-7, zh_CN.euc, zh_CN.iso2022-7, zh_TW-big5, zh_TW-euc, zh_TW-iso2022-7 **AIX platform:** ASCII-GR, CNS11643.1986-1, CNS11643.1986-2, IBM-1046, IBM-1124, IBM-1129, IBM-850, IBM-856, IBM-932, IBM-eucJP, IBM-eucKR, IBM-eucTW, IBM-sbdTW, IBM-udcJP, IBM-udcTW, ISO8859-1, ISO8859-2, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, ISO8859-7, ISO8859-8, ISO8859-9, JISX0201.1976-0, JISX0208.1983-0, KSC5601.1987-0, TIS-620, big5, ct, fold7, fold8, uucode **HP-UX platform:** roman8, iso8859_1, iso8859_2, iso8859_5, iso8859_6, iso8859_7, iso8859_8, iso8859_9, tis620, eucJP, sjis, big5, ccdc, eucKR, chinese-gb

Attribute Modifications

These operations simplify global attribute modifications by replacing or removing attributes for all entries in the LDIF input.

Table 21-4 Attribute Name Modifications Using `ldifxform`

Command	Formatting Effect
`-c suppressoptions`	Remove all options other than binary from attribute type names.
`-c tcut=`*attr*	Remove the attribute named *attr* from all entries where it is found. To remove multiple attributes, specify this command multiple times on the command-line.

Table 21-4 Attribute Name Modifications Using `ldifxform` *(Continued)*

Command	Formatting Effect
`-c tpreserve=`*attr*	Remove all attributes except for the given *attr* from all entries. To preserve multiple attributes, specify this command multiple times on the command-line.
`-c treplace=`*old*`:`*new*	Replace the *old* attribute type name with the *new* name in all entries where it occurs.

LDAP Entry Presorting

Many directory servers return search results in the order that entries were loaded into the database. By presorting a set of entries known to be static, clients can avoid having to sort results with every query.

Table 21-5 Ordering of LDAP Entries Using `ldifxform`

Command	Formatting Effect
`-c order`	Reorders all entries in the file into hierarchical order.
`-c sort=`*attr*	Sorts entries by increasing value (lowest to highest) of the given *attr* attribute. This is equivalent to alphabetical order for string-valued attributes.
`-c sort=^`*attr*	Sorts entries by decreasing value (highest to lowest) of the given *attr* attribute. This is equivalent to reverse alphabetical order for string-valued attributes.
`-c split=`*N*	Generates multiple LDIF output files that can be loaded into a server by *N* clients in parallel. The output files are named: *outputFile*`_ldifxform_`*c*`_`*n* Where: • *outputFile* is the parameter of the `-o` option and specifies a writable directory and filename prefix. • *c* is the number of components in the root DN of the LDIF file. • *n* is the number of the part, from 1 to *N*.

Directory Analysis

The `ldifxform` tool can be used to analyze the contents of the directory from which the LDIF file is extracted. The output includes a detailed count of DN structures and attribute value occurrences.

Table 21-6 Extracting Directory Statistics Using `ldifxform`

Command	Formatting Effect
`-c stats`	Generates statistical information and appends it as an LDIF comment to the end of the output file.
`-c statsonly`	Generates and outputs statistical information only. This command is not compatible with any other reformatting or LDIF transformation commands.

Command-Line Examples

The examples in this section show how to use the `ldixform` tool. These examples are based on the input file `two.ldif` detailed in Code Example 21-1.

Code Example 21-1 two.ldif Input File

```
dn: sn=Jensen,dc=example,dc=com
objectclass: top
objectclass: person
cn: Babs Jensen
sn: Jensen
telephoneNumber: 555-5550
createTimestamp: 100

dn: sn=Minsky,dc=example,dc=com
objectclass: top
objectclass: person
cn: Pete Minsky
sn: Minsky
telephoneNumber: 555-5559
modifyTimestamp: 200

dn: sn=Morris,dc=example,dc=com
objectclass: top
objectclass: person
cn: Ted Morris
sn: Morris
telephoneNumber: 555-5558
createTimestamp: 200
```

Transforming LDIF to a List

The following command will reformat the information in `two.ldif` so it is presented as a simple, hierarchical list of employees placed in order by their telephone numbers. The result appears on the standard output because no `-o` option is specified.

```
$ ldifxform -i /export/temp/two.ldif -c "tpreserve=telephonenumber" \
            -c "tpreserve=cn" -c "sort=telephonenumber" \
             -c nodn -c notypes
```

Removing the DNs saves space and makes the information more readable. The *sentinel* markers are used internally by the tool and can be edited out of the result if not needed.

Code Example 21-2 Hierarchical List of Employees and Telephone Numbers

```
version: 1
#:ordered: TRUE

objectclass: top
objectclass: sentinel

objectclass: top
objectclass: sentinel

Babs Jensen
555-5550

Ted Morris
555-5558

Pete Minsky
555-5559
```

Statistical Output

The following command shows the statistical output of the `ldifxform` tool. The description of the various counters is self-contained within the generated comments.

```
$ ldifxform -i /export/temp/two.ldif -c statsonly
```

The output includes a detailed count of DN structures and attribute value occurrences. Code Example 21-3 details the output.

Code Example 21-3 Statistics Only Output

```
# Basic statistics
#:linecount: 27
#:entrycount: 4
# Number of nonleaf entries (at least one subordinate)
#:nonleafcount: 1
# Number of leaf entries (no subordinates)
#:leafcount: 4
# Largest number of entries immediately below a nonleaf entry
#:maximmsubr: 4
# Number of levels in the DIT hierarchy
#:maxdepth: 3
# Largest number of AVAs in an RDN of an entry's DN (normally 1)
#:maxrdns: 1
# Attribute types used in the LDIF file
# e is number entries containing this attr, v is total number of
# values, l is total length, m is max length of any one value,
# s is general syntax and x is extra encoding information.
#:attrstatsinfo: t=deletetimestamp e=1 v=1 l=3 m=3 i=1 s=int
#:attrstatsinfo: t=telephonenumber e=3 v=3 l=24 m=8 i=1 s=tel
#:attrstatsinfo: t=sn e=3 v=3 l=18 m=6 i=1 s=cis x=alphanumeric
#:attrstatsinfo: t=cn e=3 v=3 l=32 m=11 i=1 s=cis x=ascii
#:attrstatsinfo: t=objectclass e=4 v=7 l=38 m=11 i=2 s=cis
#               x=alphanumeric
# Counts of values of specific attribute types
#:attrdomaininfo: t=objectclass v=1 nsTombstone
#:attrdomaininfo: t=objectclass v=3 person
#:attrdomaininfo: t=objectclass v=3 top
# Number of entries with the latest createTimestamp value (UTC)
#:lastaddcount: c=1 t=200
# Number of entries with the latest modifyTimestamp value (UTC)
#:lastmodcount: c=1 t=200
```

Chapter 22

The LDIF Merge Tool

The `mmldif` tool combines multiple LDIF (LDAP Data Interchange Format) files into a single directory hierarchy. The result is the union of all input files. This chapter provides instructions on how to use the `mmldif` tool. It contains the following sections:

- Overview
- Command Usage
- Command-Line Examples

Overview

The `mmldif` tool merges multiple LDIF files into a single directory hierarchy. The result is one output file, containing any entry whose DN appears in one or more of the input files. In a typical usage scenario, `mmldif` can be used to create an authoritative database for servers cooperating in a multi-master replication agreement. If for some reason all masters are no longer able to synchronize, the `mmldif` tool can be used to generate the master database manually.

Command Usage

The `mmldif` tool unites all input files, based on the DNs they contain: entries that exist in one or more input files are added to the hierarchy of the final output. When the same DN appears in multiple files, the tool will verify that all attribute values are identical. If the attribute values differ, the conflict is resolved by choosing the entry with the most recent time stamp. The time stamp is defined by one of the following attributes: `modifyTimestamp`, `createTimestamp`, or `deleteTimestamp`. `mmldif` also recognizes deleted entries by the `objectclass: nsTombstone` attribute

and handles them accordingly. For LDIF files to contain these special attributes, you must use the `db2ldif -r` option when extracting the entries from your directories. For more information, see "db2ldif (Export Database Contents to LDIF)" in Chapter 2 of the *Sun ONE Directory Server Reference Manual*.

Syntax

The syntax of the `mmldif` tool on the command-line takes the following form:

```
mmldif [-c] [-o output.ldif] input1.ldif input2.ldif ...
```

Where:

- *output*`.ldif` is the name of a writable file for the LDIF output.
- *inputn*`.ldif` is an LDIF input file to be merged. You must specify at least two input files.

Options

The `mmldif` options and parameters are described in Table 22-1. The `mmldif -h` command will display a usage help text that briefly describes all options.

Table 22-1 Command-Line Options for `mmldif`

Option	Parameter	Purpose
`-o`	*outputFile*	Specify the output file for the merge of all LDIF inputs. The output file is itself a complete directory hierarchy in LDIF text. When this parameter is omitted, the `mmldif` tool produces no output and reports only the number of differences between the input files.
`-c`		Write a change file containing LDIF update statements for each input file. The change file will have the same path and filename as the corresponding input file, with the addition of the suffix `.delta`. When using this option, the input files must be in a writable location. Old change files for the same input will be overwritten.
`-h`		Display the usage help text that briefly describes all options.

Command-Line Examples

The examples in this section demonstrate different types of output from the `mmldif` tool. They use as input two LDIF files: `one.ldif` and `two.ldif`. Both files are reproduced in Table 22-2 with the differences between the two formatted in **bold**.

Table 22-2 one.ldif And two.ldif Input Files

Filename: **one.ldif**	Filename: **two.ldif**
dn: sn=Jensen,dc=siroe,dc=com objectclass: top objectclass: person cn: Babs Jensen sn: Jensen telephoneNumber: 555-5550 createTimestamp: 100	dn: sn=Jensen,dc=siroe,dc=com objectclass: top objectclass: person cn: Babs Jensen sn: Jensen telephoneNumber: 555-5550 createTimestamp: 100
dn: sn=Minsky,dc=siroe,dc=com objectclass: top objectclass: person cn: Pete Minsky sn: Minsky telephoneNumber: **555-5551** createTimestamp: **100**	dn: sn=Minsky,dc=siroe,dc=com objectclass: top objectclass: person cn: Pete Minsky sn: Minsky telephoneNumber: **555-5559** modifyTimestamp: **200**
dn: sn=Rose,dc=siroe,dc=com **objectclass: top** **objectclass: person** **cn: Paula Rose** **sn: Rose** **telephoneNumber: 555-5552** **createTimestamp: 100**	**dn: sn=Morris,dc=siroe,dc=com** **objectclass: top** **objectclass: person** **cn: Ted Morris** **sn: Morris** **telephoneNumber: 555-5558** **createTimestamp: 200**
	dn: sn=Rose,dc=siroe,dc=com **objectclass: nsTombstone** **deleteTimestamp: 200**

Merge Statistics

The `mmldif` tool always displays merge statistics to the standard output. The `-o` *outputFile* parameter may be omitted when you are interested only in the number of differences between input files as in this command:

```
$ mmldif one.ldif two.ldif
```

The output reproduced inCode Example 22-1 confirms the differences between the two input files.

Code Example 22-1 Merge Statistics Output

```
start time Wed Jul 11 12:34:56 2001
entry counts: unchanged=1 changed=2 new=1 total=4
end time Wed Jul 11 12:34:56 2001
differencing took <= 1 second
```

By inspecting the LDIF files in Table 22-2, we can see the following (also detailed in Code Example 22-1):

- `unchanged=1` - There is one entry where all attributes match exactly.
- `changed=2` - Two entries with matching DNs have different attribute values.
- `new=1` - One entry is considered new as its DN does not appear in all files.
- `total=4` - The input files contain a total of four different DNs.

Merge Output File

Running the command detailed in Merge Statistics with the `-o` *outputFile* parameter will allow us to see the result of the merge.

```
$ mmldif -o merge.ldif one.ldif two.ldif
```

The output reproduced inCode Example 22-2 details the result of the union between the two input files.

- The first entry, `sn=Jensen`, remains unchanged.
- Only the telephone number is modified in the second entry, `sn=Minsky`.
- The third entry, `sn=Morris`, is counted as new because it was not present in `one.ldif`.
- The deleted entry, `sn=Rose`, is not included, although it is tallied as a change in the entry counts.

Code Example 22-2 mmldif Output File

```
start time Wed Jul 11 12:34:56 2001
entry counts: unchanged=1 changed=2 new=1 total=4
end time Wed Jul 11 12:34:56 2001
differencing took <= 1 second
```

Code Example 22-2 mmldif Output File *(Continued)*

```
$ cat merge.ldif

dn: sn=Jensen,dc=siroe,dc=com
cn: Babs Jensen
CreateTimestamp: 100
objectclass: person
objectclass: top
sn: Jensen
telephoneNumber: 555-5550

dn: sn=Minsky,dc=siroe,dc=com
cn: Pete Minsky
objectclass: person
objectclass: top
sn: Minsky
telephoneNumber: 555-5559

dn: sn=Morris,dc=siroe,dc=com
cn: Ted Morris
CreateTimestamp: 200
objectclass: person
objectclass: top
sn: Morris
telephoneNumber: 555-5558
```

Produce Change Files

To see how each input relates to the merge result, `mmldif` can generate change files. Change files contain LDIF update statements that represent the difference between the corresponding input and the merged result. Applying the change file to the directory of the corresponding input will make the directory contents equivalent to the contents of the merged result. To do this, the change file can be used as input to the `ldapmodify` command. (See Chapter 4, “The ldapmodify Tool” for more information.) LDIF update statements showing the difference between each input file and merge result can be generated by running the following command:

```
$ mmldif -o merge.ldif -c one.ldif two.ldif
```

The output reproduced inCode Example 22-3 details the changes between the two input files.

Code Example 22-3 Output of Changes

```
start time Wed Jul 11 12:34:56 2001
entry counts: unchanged=1 changed=2 new=1 total=4
end time Wed Jul 11 12:34:56 2001
differencing took <= 1 second
```

Code Example 22-4 shows how to bring the contents of `one.ldif` up to date with the result of the merge. The changes detailed are:

- Modify the entry for `sn=Minsky` by:
  - Deleting the `createTimestamp` attribute. (`mmldif` removes timestamp attributes from modified entries.)
  - Replacing the value of the `telephoneNumber` attribute.
- Delete the entire entry for `sn=Rose`.
- Add the entire new entry for `sn=Morris`.

In addition to `one.ldif.delta`, the file `two.ldif.delta` is also generated but it is empty (zero length) because `two.ldif` contains all of the most recent modifications between the two input files.

Code Example 22-4 one.ldif.delta

```
$ cat one.ldif.delta

dn: sn=Minsky,dc=siroe,dc=com
changetype: modify
delete: createTimestamp
-

replace: telephoneNumber
telephoneNumber: 555-5559
-

dn: sn=Rose,dc=siroe,dc=com
changetype: delete
dn: sn=Morris,dc=siroe,dc=com
changetype: add
cn: Ted Morris
createTimestamp: 200
objectclass: person
objectclass: top
```

Code Example 22-4 one.ldif.delta *(Continued)*

```
sn: Morris
telephoneNumber: 555-5558
```

Supposing that `one.ldif` represents the contents the directory on host `one`, port `389`, the following command will update those contents so that they are equivalent to `merge.ldif`:

```
$ ldapmodify -h one -D "bindDN" -w bindPassword \
             -f one.ldif.delta
```

This command uses the credentials of the *bindDN* to access the directory and assumes that user has permission to modify entries. See Chapter 4, “The ldapmodify Tool,” for more information.

Chapter 23

The LDAP Compare and Modify Tool

The `ldiffer.pl` tool detects differences between two LDAP directories and performs modifications accordingly. This chapter provides instructions on how to use the `ldiffer.pl` tool. It contains the following sections:

- Overview
- Command Usage
- Output Files
- Configuration File
- Working With ldifxform

Overview

The `ldiffer.pl` tool is a Perl script that detects differences between two LDAP directories and can perform modifications accordingly. Comparisons are based on attribute values so that directories with different DN structures and different schemas may be compared. When the tool detects comparable entries in each directory, it can be configured to add or modify attributes in either entry. The value of new or modified attributes is given as a regular expression involving other attribute values. This flexibility can be used to migrate data to a new schema or to import specific data from one directory into another. The DSRK includes the tool in the *DSRK_base*`/perl` directory.

NOTE This script requires Perl version 5.005_03 or later. See “Third-Party Sources of Information” on page 32 for links to Perl resources.

Using the Script

`ldiffer.pl` interacts with two directories. One is called *sequential* and can be considered the source of information. The other is called *random* and can be considered the target to be modified. However, the tool also allows modifications to be made in the sequential directory. Entries from each directory are matched based on equality between the value of one or more key attributes, allowing you to match entries that do not necessarily have the same DN. Because only attribute values are compared, key attributes may have different names in each directory. (The bind credentials, the base DN, and the filter for each search are specified separately, allowing arbitrarily different sets of entries to be compared.)

Within a pair of matched entries, the tool compares the values of one or more attributes using a *regular expression (*a string of characters that defines a pattern used to search for a matching string). A discrepancy between any of them will trigger the tool to perform a set of update actions. An *update action* involves modifying an attribute in one of the entries based on the value of an attribute in its matched entry. For example, an administrator may wish to migrate information from an old directory under the suffix `o=example.com,c=us` to a new directory under `dc=example,dc=com`.. The key attribute might be employee login names; this attribute would be invariant. `ldiffer.pl` could then compare an attribute such as `fax` in a custom schema to `facsimileTelephoneNumber` in the standard schema of the new directory. The value of `facsimileTelephoneNumber` could then be updated with the value of the `fax` attribute or with a new area code. When matching entries are not found, the tool may also be configured to delete the entry in the source directory, add it to the target directory, or both.

NOTE The `ldiffer.pl` script uses `ldapsearch` to extract a set of entries from both directories and `ldapmodify` to modify entries. See Chapter 3, “The ldapsearch Tool” and Chapter 4, “The ldapmodify Tool” for more information on these tools.

Customizing the Script

If you customize the `ldiffer.pl` script for added functionality, we encourage you to share your work with other LDAP users. Please post a message to the `iplanet.server.idsrk` public newsgroup with your ideas or your code.

Command Usage

Due to the flexibility it offers, the `ldiffer.pl` script requires many parameters to describe the modifications of one or both target directories. These parameters are defined in a configuration file that is named on the command-line. The configuration file is discussed in “Configuration File” on page 274.

The syntax of `ldiffer.pl` on the command-line takes the following form:

```
ldiffer.pl configFile
```

Where *configFile* is the name of a file that contains all of the information for binding to the directories, performing comparisons, and making any potential modifications. The file and its format is described in “Configuration File”.

NOTE You will need to specify values for all of the parameters in the configuration file before running `ldiffer.pl`. Due to the potential complexity of comparisons and updates, we recommend testing your configuration files on mock directories with the debugging parameter turned on.

Output Files

All output from the script is written to log files with names and locations based on information in the configuration file. The following sections detail these log files.

configFile.timestamp.action.log

configFile`.`*timestamp*`.action.log` displays the comparisons, additions, and modifications that were performed in a given run. It also gives a count of the differences and lists the entries that were not successfully matched.

configFile.timestamp.debug.log

configFile`.`*timestamp*`.debug.log` records commands that were executed while comparing and modifying the directories. This file is created only if the debugging parameter is specified.

Configuration File

The `ldiffer.pl` script requires a configuration file that defines both the sequential and the random directories. The DSRK installation includes an example configuration file named `ldiffer-sample.config`. The file is located in *DSRK_base*`/perl` and reproduced in Code Example 23-1.

Code Example 23-1 `ldiffer-sample.config` Sample Configuration File

```
# Global parameters
ldapsearch:/opt/iPlanet/bin/dsrk52/ldapsearch
ldapmodify:/opt/iPlanet/bin/dsrk52/ldapmodify
debug:1

# the delimiter should be a character that will not
# occur in the values of key attributes to match
delimiter:!

# The "sequential" directory can be considered the source
sqn_h:legacy.example.com
sqn_p:389
sqn_D:cn=directory manager
sqn_w:bindPassword
sqn_b:o=example.com,c=us
sqn_filter:login=*
sqn_key_attrs:login
sqn_comp_attrs:dn
sqn_add_attrs:branch;l;
sqn_mod_attrs:
sqn_del_unmatched:0

# The "random" directory can be considered the target
rnd_h:ldap.example.com
rnd_p:389
rnd_D:cn=directory manager
rnd_w:bindPassword
rnd_b:dc=example,dc=com
rnd_filter:uid=*
rnd_key_attrs:uid
rnd_comp_attrs:dn
rnd_add_attrs:
rnd_mod_attrs:mail;email;,cn;fullName;,sn;fullName;s/.*\s//
rnd_add_unmatched:0
```

Configuration File Syntax

The configuration file uses the following syntax:

```
# line of comment
parameter:value
...
```

The parameters may be given in any order, but are usually grouped according to the directory they affect. The file contains three sections grouped as global, sequential and random parameters.

Global Parameters

The global parameters in Table 23-1 affect the behavior of the tool.

Table 23-1 Global Parameters in the `ldiffer.pl` Configuration File

Parameter	Purpose
`ldapsearch`	Specify the path for running the `ldapsearch` tool. If you installed the Sun™ ONE DSRK in the default location, this parameter should be set to `/opt/iPlanet/bin/idsrk/ldapsearch`.
`ldapmodify`	Specify the path for running the `ldapmodify` tool. If you installed the Sun ONE DSRK in the default location, this parameter should be set to `/opt/iPlanet/bin/idsrk/ldapmodify`.
`debug`	Indicate the use of the debugging log. The possible values are `0` for no logging or `1` for creating the log file.
`delimiter`	Set the character used internally to delimit key attribute values. You should set this parameter to a character that does not occur in the attribute values of your directories, for example `!`.

Sequential and Random Parameters

All sequential and random parameters have one of the following prefixes:

- `sqn_` for parameters that apply to the *sequential* (or source) directory.
- `rnd_` for parameters that apply to the *random* (or target) directory.

Connection Parameters

The connection parameters determine the LDAP directories to operate upon and give the bind credentials for accessing them. They are detailed in Table 23-2.

Table 23-2 Connection Parameters in the `ldiffer.pl` Configuration File

Parameter	Purpose

`sqn_h`
`sqn_p`
`rnd_h`
`rnd_p`
	Specify the host name or IP address and port number of each LDAP directory.

`sqn_D`
`sqn_w`
`rnd_D`
`rnd_w`
	Give the bind DN and password needed to access each directory. These parameters may be left blank for anonymous binding. The bind credentials must have permissions for both searching the directories and modifying entries, if necessary. Also note that the bind DN may influence the results of searches when determining the set of entries to compare.

Comparison Parameters

The comparison parameters determine the set of entries to compare, how to match entries from each directory, and the attribute values that will trigger actions. They are detailed in Table 23-3.

Table 23-3 Comparison Parameters in the `ldiffer.pl` Configuration File

Parameter	Purpose

`sqn_b`
`sqn_filter`
`rnd_b`
`rnd_filter`
	These parameters determine the set of entries that will be compared from each directory. Their values are not necessarily identical, as the `ldiffer.pl` tool allows you to match entries based on any attribute values, not only the DN of an entry. Specify the base DN and filter string used to search each directory. The filter string should be an RFC 2254-compliant LDAP search filter, for example: `(uid=*)`. For more information, see "LDAP Search Filters" in Chapter 4 of the *Sun ONE Directory Server Getting Started Guide*.

`sqn_key_attrs` `rnd_key_attrs`	Specify the names of the key attributes whose values must be equal in order to match an entry from each directory. The parameters may contain multiple attribute names in a comma separated list. The lists may contain different attribute names, but they must contain the same number of names. In order for two entries to match, all attribute values must match in the order that the attribute names are given in their respective list. To match entries based on their DN, specify the dn attribute in both parameters.

Table 23-3 Comparison Parameters in the `ldiffer.pl` Configuration File *(Continued)*

Parameter	Purpose
`sqn_comp_attrs` `rnd_comp_attrs`	Specify the names of the comparison attributes for entries from each directory. These are comma separated lists containing an equal number of attribute names. When two entries match according to the key attributes, the values of the comparison attributes are compared in the order that the attribute names are given in their respective list. If any one of the comparison attribute values differ in the pair, the two entries will be modified according to the action parameters. To perform a DN-based comparison, specify the `dn` attribute in both lists. If any one of the attributes does not exist in its respective entry, the comparison will fail but no action will be triggered. If you wish to trigger an action for every pair of matching entries, you must specify a single attribute name in each list parameter that you know will have a value and be different on each entry.

Action Parameters

The action parameters in the following table determine what modifications will be made when matching entries fail the comparison or when entries fail to match. These parameters are complex lists of:

Statement1`,`*Statement2*`, . . .`

Where each *Statement* has the following syntax:

AttributeName`;`*otherAttribute*`;`*regularExpression*

The *regularExpression* must follow the syntax for Perl regular expressions or they may also be omitted, but the punctuation of the list must be respected, as in:

```
givenName;firstName;,sn;lastName;
```

Table 23-4 Action Parameters in the `ldiffer.pl` Configuration File

Parameter	Purpose
`sqn_add_attrs` `rnd_add_attrs`	Specify the attributes to add to each of the matched entries and how to determine their initial value. These parameters are complex lists of the format previously described. Each *Statement* will create a new attribute with the given *AttributeName* in the entry of the corresponding directory (either `sqn` or `rnd`). The initial value given to the attribute will be the value of the *otherAttribute* in the matching entry (in the other directory), after applying the *regularExpression*. To create multivalued attributes, specify multiple add statements with the same *AttributeName*.

Table 23-4 Action Parameters in the `ldiffer.pl` Configuration File *(Continued)*

Parameter	Purpose
`sqn_mod_attrs` `rnd_mod_attrs`	Specify the attribute values to modify and how to determine their new value. These parameters have the same complex list format as described previously. Each *Statement* will modify the value of the given *AttributeName* in the entry of the corresponding directory (either `sqn` or `rnd`). The new value will be that of the *otherAttribute* in the matching entry (in the other directory), after applying the *regularExpression*.

All actions specified by the parameters in Table 23-4 will be triggered when a pair of matching entries fail the comparison. Thus, you may configure `ldiffer.pl` to add attributes or modify entries in both directories. Leave parameters blank if that specific action is not required. For example, if you do not wish to modify the source `sqn` directory, do not define the `sqn_add_attrs` or `sqn_mod_attrs` parameters.

Unmatched Entry Parameters

The tool currently offers less flexibility for entries that do not match. After pairing up all entries in the chosen set from each directory, `ldiffer.pl` can only determine those entries from the source `sqn` directory that do have a match. These entries may optionally be removed from the `sqn` directory, added to the `rnd` directory, or both, according to the parameters in Table 23-5.

Table 23-5 Unmatched Entry Parameters in the `ldiffer.pl` Configuration File

Parameter	Purpose
`rnd_add_unmatched`	Specify whether unmatched entries from the source `sqn` directory will be added to the target `rnd` directory. The value of this flag is either `0` for no additions or `1` for adding all unmatched entries. When entries are added, there is no mechanism to modify their attributes or attribute values. Therefore, when this flag is true (`1`), the target `rnd` directory must support the same DN hierarchy and the same schema as the source `sqn` directory.
`sqn_del_unmatched`	Specify whether unmatched entries are deleted from the source `sqn` directory. The value of this flag is either `0` to leave the `sqn` directory unchanged or `1` to remove all unmatched entries.

Configuration File Use Scenario

The configuration file reproduced in Code Example 23-1 on page 274 might be used in a hypothetical scenario where `example.com` has undergone geographic expansion and created a new LDAP directory using the standard schema. Some data has already been added to the new directory, such as employee `uid` and location (`l`). The rest of the needed information can be extracted from the legacy directory that has a custom schema. Running the `ldiffer.pl` tool with this configuration file will match entries based on the `uid` attribute. Then, data from the legacy directory can be used to set attributes with standard names (`mail`, `cn`, and `sn`) in the new directory. If these attributes do not exist in the new directory, they will be created. The value of the surname (`sn`) attribute is assumed to be the second word of the legacy `fullName` attribute, as extracted by the Perl regular expression `s/.*\s//`. However, the legacy system is still in use by the human resources department, and now that `example.com` has several offices, it also needs to update the legacy directory. The tool will copy the employees' new locations into a new attribute called `branch`.

After the migration, `example.com` might also wish to keep information in the legacy directory up to date, assuming that the latest modifications occur only in the new directory. The `ldiffer.pl` tool can be used to automate the transfer of information back into the schema of the legacy directory. Another configuration file would be needed for this scenario, in which the attribute values in the new `rnd` directory would be used to modify attributes in the legacy `sqn` directory.

Working With ldifxform

Updating or creating new directories can be highly automated through the use of the `ldifxform` and `ldiffer.pl` tools. (For more information on these tools, see Chapter 21, "The LDIF Transformation Tool.") Together, they can modify extracted information for complex operations such as porting existing data to a new schema. However, careful planning is necessary to perform a complicated sequence of transformations. Directory modifications of this scope will require special considerations such as turning off schema checking and disallowing other read and write requests while the data is in an intermediate state.

Part 6

Maintenance and Debugging Tools

Chapter 24

The Log Analyzer Tool

The `logconv.pl` tool analyzes the access logs of Sun™ ONE Directory Server. It extracts usage statistics and counts the occurrences of significant events. This chapter provides instructions on how to use `logconv.pl`. It contains the following sections:

- Overview
- Command Usage
- Command-Line Examples

Overview

`logconv.pl` is a Perl script that analyzes the access logs of Sun ONE Directory Server in order to extract usage statistics and count the occurrences of significant events. It is compatible with log formats from Directory Server 3.*x*, 4.*x*, and 5.*x*.

NOTE Some information extracted by the `logconv.pl` script is available only in Sun ONE Directory Server 5.*x* logs thus, the corresponding values will be zero when analyzing logs from other versions. In addition, some information will only be present in the logs if verbose logging is enabled in your directory server. For more information, see "nsslapd-accesslog-level" in Chapter 4 of the *Sun ONE Directory Server Reference Manual*.

The DSRK includes the tool in the *DSRK_base*`/perl` directory.

NOTE This script requires Perl version 5.005_03 or later. See "Third-Party Sources of Information" on page 32 for links to Perl resources.

Statistics for Display

The `logconv.pl` tool displays three types of statistics that administrators will find useful for monitoring and optimizing directory usage. They are:

- Simple counts of events such as the total number of binds and the total number of searches.
- Lists of the most frequently occurring parameters in LDAP requests. For example, lists of the top ten bind DNs, base DNs, filter strings, and attributes returned. Because they are computation intensive, these lists are optional; specify only the command-line options for those you need.
- A count of occurrences of any given error code, along with a listing of the corresponding error messages in the log file.

Tool Performance

The following issues will affect the output and performance of this tool:

- Some data extracted from logs depends on connection and operation numbers that are reset and no longer unique after a server restarts. Therefore, to obtain the most accurate counts, the logs to be analyzed should not span the restart of the directory server.
- Due to changes in access logs formats in Directory Server 5.0 that also affect operation numbers, the tool will be more accurate on 5.*x* logs when processing large amounts of access logs.
- For performance reasons, it is not recommended to run more than one gigabyte of access logs through the script at any one time.

Customizing the Script

If you customize the `ldiffer.pl` script for added functionality, we encourage you to share your work with other LDAP users. Please post a message to the `iplanet.server.idsrk` public newsgroup with your ideas or your code.

Command Usage

`logconv.pl` analyzes access logs by extracting usage statistics and counting the occurrences of significant events. The tool will extract the following information from access logs:

- Number of restarts
- Total number of connections
- Total operations requested
- Total results returned
- Results to requests ratio
- Number of searches, modifications, adds, deletes, and modified RDNs
- Virtual list view (VLV) operations
- VLV unindexed searches
- Server-side sorting operations
- SSL connections
- Performance lowering operations such as entire database searches or unindexed searches (details optional)
- File descriptors taken and returned
- Highest file descriptor taken
- Disruptions such as broken pipes, connections reset by peer or unavailable resources (and detail)
- Bind information such as total binds, types of binds, expired password logins, and failed binds
- Most frequent occurrence lists (optional) for error and return codes, failed logins, connection codes, client IP addresses and connection codes, bind DNs, base DNs for searching, search filters, etimes (elapsed operation time) and longest etimes, nentries (number of entries in result) and largest nentries, extended operations (DS 5.*x* only), most requested attributes (DS 5.*x* only), and abandoned operation details (DS 4.15)
- Recommendations (optional)

NOTE For Sun ONE Directory Server 5.*x* logs only, the following will also be extracted:

- Persistent searches
- Internal operations (with verbose logs)
- Entry operations (with verbose logs)
- Extended operations
- Abandoned requests
- Smart referrals received (verbose logs)

Syntax

The syntax of `logconv.pl` on the command-line takes the following form:

```
logconv.pl [options] [-efcibaltnxgju | -E errorCode ] accessLog ...
```

Where:

- *options* and `[-efcibaltnxgju]` are the command-line options described in "Options."
- *errorCode* will display a report only on occurrences of the given error number.
- *accessLog* is the name of a file that contains the access log of your Directory Server. You may use wildcards in the filename or specify multiple filenames. However, the statistics are computed over the set of all logs, so all logs should pertain to the same directory server. The tool will ignore any file with the name `access.rotationinfo`.

Options

The `logconv.pl` command-line options are described in Table 24-1. Regardless of the order of options on the command-line, the lists will appear in the output in the order they are listed in this table. Use the `-V` option to display all optional output. Also, use the `-s` *number* option to control the length of these lists. The `logconv.pl -h` command will display the usage help text that briefly describes all options.

NOTE The parameters without a preceding dash (`-`) at the end of the table will enable the optional lists of occurrences. Specify only those you need to limit the output and improve execution speed. You may specify any number of these parameters in any order, but they must all be given together as a single option on the command-line, for example: `-abcefg`.

Table 24-1 Command-Line Options for `logconv.pl`

Option	Parameter	Purpose
`-d`	*mgrDN*	Specify the DN (distinguished name) of the directory manager in the logs being analyzed. This allows the tool to collect statistics for this special user. The *mgrDN* parameter should be given in double quotes (`""`) for the shell. When this parameter is omitted, `logconv.pl` will use the default manager DN of Sun ONE Directory Server: `"cn=directory manager"`.
`-N`		Enable DNS lookup on IP addresses found in the log file so that machine names appear in the output instead.
`-X`	*IPaddress*	Specify the IP address of a client to exclude from the statistics. This client will not appear in lists of IP addresses (the `i` flag), and the connection codes it generates will not be tallied in the total connections (default statistic) nor in the connection code details (the c flag). For example, you may wish to ignore the effect of a load balancer that connects to the directory server a regular intervals. This option may be repeated to exclude multiple IP addresses.
`-v`		Display the version number of the `logconv.pl` script.
`-h`		Display the usage help text that briefly describes all options.
`-E`	*errorCode*	Display only the information about the given, numeric error code. When using this option, the tool will not display the summary of all log information, it will only give the count and the list of the given error's occurrences. **This option is incompatible with all of the following options.**
`-V`		Enable the most verbose output. With this option, `logconv.pl` will compute and display all of the optional lists described below.
`-s`	*number*	Specify the number of items in each of the list options below. The default is `20` when this parameter is omitted. For example, `-s 10 -i` will list the ten client machines that access the server most often. This parameter applies to all lists that are enabled.
`e`		List the most frequent error and return codes.
`f`		List the bind DNs with the most failed logins (invalid password).
`c`		List the number of occurrences for each type of connection code.
`i`		List the IP addresses and connection codes of the clients with the most connections. This option helps to detect clients that may be trying to compromise security.
`b`		List the most frequently used bind DNs.
`a`		List the most frequent base DNs when performing operations.
`l`		List the most frequently used filter strings for searches.

Table 24-1 Command-Line Options for `logconv.pl` *(Continued)*

Option	Parameter	Purpose
`t`		List the longest and most frequent `etimes` (elapsed operation time).
`n`		List the largest and most frequent `nentries` (entries per result).
`x`		List the number and OID of all extended operations (DS 5.*x* only).
`r`		List the names of the most requested attributes (DS 5.*x* only).
`g`		List the details of all abandoned operations.
`j`		Give recommendations based on data collected from the log file.
`u`		Give operation details about unindexed searches.

Command-Line Examples

Following are two examples showing how the `logconv.pl` tool can be used.

Error Code Listing

The following command shows how to track error code listings. It defines error code 49 as the one to track.

```
$ perl logconv.pl -N -E 49 logs/access
```

Code Example 24-1 shows the output of the specific error code (`-E 49`), corresponding to a failed login with a bad password. The listing contains machine names (`-N`) instead of IP addresses to make it more readable. The number in the first column is the number of repeated occurrences of the same error message.

Code Example 24-1 Error Code Listing Sample Output

```
Error (49) Count:  5
Parsing & Sorting...
Number  Operation  Client                  Object
=================================================================
3          BIND    (test.example.com)        cn=Directory Manager
1          BIND    (localhost)             cn=Directory Manager
1          BIND    (test.example.com)        uid=bjensen
```

Verbose Output

The following command shows how to record a verbose output. The `logconv.pl` tool will read all access logs in the logs directory (ignoring `access.rotationinfo` files). After processing the log files, it displays all access statistics and event counters and shows the lists of most frequent connection and operation values, with the top 10 in each category (`-s 10`). It ends with a set of general recommendations triggered by certain values or events.

NOTE There are many possible recommendations depending on the statistics and occurrences of certain events. The recommendations are based on general administration guidelines and should be adapted to fit the actual usage of your directory server.

```
$ perl logconv.pl -V -s 10 \
                  /usr/iplanet/servers/slapd-serverID/logs/access*
```

Code Example 24-2 Sample Verbose Output for logconv.pl

```
verbose output enabled
Log Analyzer 4.11
Initializing Variables...
Processing 3 Access Log(s)...
access (Total Lines: 5870)
        1000 Lines Processed
        2000 Lines Processed
        3000 Lines Processed
        4000 Lines Processed
        5000 Lines Processed
*       5870 Lines Processed      Total Lines Processed:  5870
access.20010713-130613 (Total Lines: 7912)
        1000 Lines Processed
        2000 Lines Processed
        3000 Lines Processed
        4000 Lines Processed
        5000 Lines Processed
        6000 Lines Processed
        7000 Lines Processed
*       7912 Lines Processed      Total Lines Processed: 13782
access.20010714-150617 (Total Lines: 6338)
        1000 Lines Processed
        2000 Lines Processed
        3000 Lines Processed
        4000 Lines Processed
        5000 Lines Processed
        6000 Lines Processed
*       6338 Lines Processed      Total Lines Processed: 20120
* Total Lines Analyzed: 20120
```

Code Example 24-2 Sample Verbose Output for logconv.pl *(Continued)*

```
----------- Access Log Output ------------
Start of Log:  18/Jul/2001:13:08:18
End of Log:    18/Jul/2001:17:05:07
Restarts:                      1
Total Connections:              4002
Total Operations:              14818
Total Results:                 14908
Overall Performance:           100.6%
Searches:                      4354
Modifications:                 27
Adds:                          26
Deletes:                       30
Mod RDNs:                      0
5.x Stats
Persistent Searches:           1
Internal Operations:           0
Entry Operations:              0
Extended Operations:           6935
Abandoned Requests:            29
Smart Referrals Received:      0
VLV Operations:                49
VLV Unindexed Searches:        49
SORT Operations:               44
SSL Connections:               0
Entire Search Base Queries:    3912
Unindexed Searches:            1
  Unindexed Search #1
  -  Date/Time:              18/Jul/2001:13:33:19
  -  Connection Number:      2926
  -  Operation Number:       1
  -  Etime:                  0
  -  Nentries:               4001
  -  IP Address:             192.18.122.229
  -  Bind DN:                cn=directory manager
  -  Search Filter:          (objectclass=*)
FDs Taken:                     3448
FDs Returned:                  3446
Highest FD Taken:              89
Broken Pipes:                  0
Connections Reset By Peer:     0
Resource Unavailable:          1
  -  1    (T1) Idle Timeout Exceeded
Binds:                         3446
Unbinds:                       3438
 LDAP v2 Binds:                1
 LDAP v3 Binds:                3445
 Expired Password Logins:      0
 SSL Client Binds:             0
 Failed SSL Client Binds:      0
 SASL Binds:                   1
  1     DIGEST-MD5
 Directory Manager Binds:      16
 Anonymous Binds:              1
 Other Binds:                  3429
```

Code Example 24-2 Sample Verbose Output for logconv.pl *(Continued)*

```
----- Errors -----
err=0                  14737    Successful Operations
err=32                    75    No Such Object
err=12                    62    Unavailable Critical Extension
err=10                     3    Referral Received
err=49                     1    Invalid Credentials (Bad Password)
err=65                     1    Objectclass Violation

----- Top 10 Failed Logins ------
1           uid=rmanager,cn=config

----- Total Connection Codes -----
U1                     3437    Cleanly Closed Connections
B1                        8    Bad Ber Tag Encountered
T1                        1    Idle Timeout Exceeded

----- Top 10 Clients -----
Number of Clients:  2
3440   123.456.789.001
                  3429 -  U1   Cleanly Closed Connections
                     8 -  B1   Bad Ber Tag Encountered
                     1 -  T1   Idle Timeout Exceeded
8      127.0.0.1
                     8 -  U1   Cleanly Closed Connections

----- Top 10 Bind DN's -----
Number of Unique Bind DN's: 8
3422            uid=rmanager,cn=config
14              cn=dm
5               uid=aa,cn=config
1               Anonymous Binds
1               uid=rmanager
1               cn=dma,cn=config
1               dc=dm
1               cn=dma

----- Top 10 Search Bases -----
Number of Unique Search Bases: 73
3519            root dse
256             ou=people,dc=example,dc=com
82              cn=ldbm database, cn=plugins, cn=config
57              cn=monitor
51              dc=example,dc=com
48              cn=config
30              cn=mapping tree,cn=config
28              cn=Babs Jensen,ou=peopled,c=example,dc=com
22              cn=plugins,cn=config
20              cn=features,cn=config

----- Top 10 Search Filters -----
Number of Unique Search Filters: 31
3502            (objectclass=*)
408             (|(objectclass=*)(objectclass=ldapsubentry))
119             (uid=*)
88              (objectclass=nsbackendinstance)
```

Code Example 24-2 Sample Verbose Output for logconv.pl *(Continued)*

```
6               (nsslapd-backend=userroot)
6               (nsslapd-plugintype=database)
4               (uid=bjensen)
4               (objectclass=subschema)
4               (objectclass=nsindex)
3               (cn=config)

----- Top 10 Most Frequent etimes -----
14634           etime=0
229             etime=1
9               etime=2
3               etime=7
1               etime=8
1               etime=3
1               etime=4
1               etime=5

----- Top 10 Longest etimes -----
etime=8         1
etime=7         3
etime=5         1
etime=4         1
etime=3         1
etime=2         9
etime=1         229
etime=0         14634

----- Top 10 Largest nentries -----
nentries=25                    5
nentries=11                    2
nentries=10                    2
nentries=9                     2
nentries=8                     1
nentries=5                     1
nentries=4                    16
nentries=3                    62
nentries=2                    37
nentries=1                  3986
nentries=0                   239

----- Top 10 Most returned nentries -----
3986            nentries=1
239             nentries=0
62              nentries=3
37              nentries=2
16              nentries=4
5               nentries=25
2               nentries=10
2               nentries=11
2               nentries=9
1               nentries=5

----- 5.x Extended Operations -----
3454    2.16.840.1.113730.3.5.3    Start Replication Request
                                     (incremental update)
```

Code Example 24-2 Sample Verbose Output for logconv.pl *(Continued)*

```
3438    2.16.840.1.113730.3.5.5    End Replication Request
                                   (incremental update)
43      2.16.840.1.113730.3.5.6    Replication Entry Request

----- Top 10 Most Requested Attributes -----
3420        supportedControl
3420        supportedExtension
360         All Attributes
341         numSubordinates
328         objectClass
315         nsAccountLock
144         nsBackendSuffix
104         nsslapd-suffix
36          dn
32          cn

----- Abandon Request Stats -----
 - SRCH conn=2 op=10 msgid=1092 client=127.0.0.1
 - BIND conn=2 op=0 msgid=1119 client=127.0.0.1

----- Recommendations -----
 1.  You have unindexed searches, this can be caused from a search on a
unindexed attribute, or your returned results exceeded the allidsthreshold.
Unindexed searches are not acceptable, please make any configuration changes
necessary to resolve these searches!
 2.  You have some connections that are are being closed by the idletimeout
setting.  You may want to increase the idletimeout if it is set low.
 3.  You have a high number of searches that query the entire search base.
Although this is not necessarily bad, it could be resource intensive if the
search base contains many entries.
```

The Replication Checker Tool

The `replcheck.pl` tool determines whether two or more replicated Sun™ ONE Directory Servers are synchronized. This chapter provides instructions on how to use `replcheck.pl`. It contains the following sections:

- Overview
- Tool Dependencies
- Command Usage
- Return Values
- Command-Line Example

Overview

The `replcheck.pl` tool is a Perl script used to analyze replicated directory servers. It determines whether two or more replicated Directory Servers are synchronized; it also reports which of the replicas is not up-to-date. The script accesses internal attributes to report on the status of the replicated sub-tree. The DSRK includes the tool in the *DSRK_base*`/perl` directory.

NOTE The `replcheck.pl` tool can only determine replication status for versions 5.x of the Sun ONE Directory Server.

Tool Dependencies

The `replcheck.pl` tool is dependent upon the PerLDAP libraries provided with Directory Server 5.*x*. PerLDAP is a third-party tool that has libraries of Perl routines for accessing a directory server and performing LDAP operations. In order to use the PerLDAP libraries you must build and install this tool. Instructions on this are described in Chapter 31, "Unsupported Utilities."

CAUTION By default, the first line of the `replcheck.pl` script contains:

```
#!/usr/iplanet/servers/bin/slapd/admin/bin/perl
```

You should modify this first line if you did not install Directory Server in the default location on your machine. Alternately, if you do not have Directory Server installed on your machine, modify this line to use the libraries installed in the unsupported directory of the Directory Server Resource Kit:

```
#!/opt/iPlanet/unsupported/perLdap
```

Command Usage

The syntax of `replcheck.pl` on the command-line takes the following form:

`replcheck.pl [-w` *DirectoryManagerPassword*`] -b` *baseDN host1*`:`*port1 host2*`:`*port2* `...`

Where:

- *DirectoryManagerPassword* is the password for the `"cn=Directory Manager"` common to all replicas. If this password is not identical on all replicas, omit this option to be prompted interactively for each password.
- *baseDN* is root of the subtree for which you wish to know the replication status.
- *hostn*`:`*portn* is the hostname and port number for each of the replicated Directory Servers. If `:`*port* is omitted, port 389 is used by default. You must specify at least two *hostn*`:`*portn* pairs, and you may specify more replicas as you wish.

The `replcheck.pl -H` command will display the usage help text that gives the command syntax.

Return Values

The `replcheck.pl` script returns `0` if all replicas are synchronized, and a positive integer otherwise. This allows the script to be used in other administrative scripts if, for example, there is a need to perform automated diagnostics.

If a replica is not in sync, the script will return its 0-based index (*n*-1) in the list of hosts on the command-line. For example, if the replica on *host2* is not up to date, the script will return `1`.

Command-Line Example

The sample reproduced in Code Example 25-1 shows the output for a comparison of an internal attribute, the *replication update vector*, that stores the replication status. The syntax used for this is:

```
$ perl -I installDir replcheck.pl -w password -b "dc=example,dc=com" \
                                  master:1389 ro-hub:1389
```

In this scenario, the command checks the replication status for the entire replicated tree under the `"dc=example,dc=com"` suffix, between a supplier replica hosted by `master` and a read-only consumer hosted by `ro-hub`.

Code Example 25-1 Sample Output for replcheck.pl

```
Connected to master
Connected to ro-hub
-------------------------

Getting replication update vector for master replica of dc=example,dc=com
=> {replica 1 ldap://master.example.com:1389} 3c63e0b0000000010000
3c63f44b000100010000
=> {replica 3 ldap://ro-hub.example.com:1389}
=> {replicageneration} 3c63da79000000030000
Getting replication update vector for ro-hub replica of dc=example,dc=com
=> {replica 1 ldap://master.example.com:1389} 3c63e0b0000000010000
3c63f443000100010000
=> {replica 3 ldap://ro-hub.example.com:1389}
=> {replicageneration} 3c63da79000000030000
****

<{replica 1 ldap://master.example.com:1389} 3c63e0b0000000010000
3c63f44b000100010000>
----
<{replica 1 ldap://master.example.com:1389} 3c63e0b0000000010000
3c63f443000100010000>
****************
```

Code Example 25-1 Sample Output for replcheck.pl

```
Replication Update Vector for ro-hub doesn't match with master
Servers are not in sync
$ echo $?    ## show return value
1
```

The first part of the output (up to the **** separator) displays the replication update vector in each replica. The second part compares all vector elements for each pair of replicas being considered and displays those that do not match. These elements encode information about the latest replication operations, and any differences in this pair of values indicates that the replicas are not up-to-date.

In this example the two elements differ slightly, and the tool displays the unmatched vector elements and human-readable explanation. Finally, the return value shows that the `replcheck.pl` tool reported that the second host on the command-line, `ro-hub`, is out-of-date.

NOTE The `-I` *installDir* option in the example is a standard Perl option and is discussed in the Perl man page.

Chapter 26

The Schema Migration Tool

The `migrateSchemaTo5.pl` tool helps automate the process of updating a custom Sun™ ONE schema for deployment with Sun ONE Directory Server 5.*x*. This chapter provides instructions on how to use `migrateSchemaTo5.pl`. It contains the following sections:

- Overview
- Command Usage

Overview

Most custom schema files defined for, and used in, version 4.x Sun ONE Directory Servers can be fully updated using the `migrateSchemaTo5.pl` tool, thus requiring no manual editing.

TIP For more information about migrating to versions 5.*x* Directory Servers, see Chapter 6, "Migrating from Previous Versions" in the *Sun ONE Directory Server Installation and Tuning Guide*.

The DSRK includes the tool in the DSRK_base/`perl` directory.

NOTE This script requires Perl version 5.005_03 or later. See "Third-Party Sources of Information" on page 32 for links to Perl resources.

Command Usage

The `migrateSchemaTo5.pl` tool is designed to convert all of the custom schema files used by your previous versions of Directory Server to the format required by versions 5.*x*. For example, all attribute and object class definitions now require a value for the `X-ORIGIN` field, and this tool will generate one automatically using either a default value or a user-specified value.

Syntax

The syntax of `migrateSchemaTo5.pl` on the command-line takes the following form:

`migrateSchemaTo5.pl -o` *4xInstancePath* `-s` *newSchemaFile* `[` *options* `]`

Where:

- *4xInstancePath* is the location of the old 4.*x* server installation. The tool will automatically locate all schema files defined in the `slapd.conf` file under the `userat` and `useroc` headings.

NOTE Schema files specified in `INCLUDE` statements with the 4.*x* `slapd.conf` file are not converted automatically. You must convert these files separately using the `-i` option described in Table 26-1.

- *newSchemaFile* is the name of the file where the equivalent 5.*x* schema will be written.
- *options* are the command-line options and their parameters described in "Options."

Options

The `migrateSchemaTo5.pl` options and parameters are described in Table 26-1.

Table 26-1 Command-Line Options for `migrateSchemaTo5.pl`

Option	Parameter	Purpose
`-o`	*4xInstancePath*	Specify the path of the 4.x Directory Sever instance that uses a custom schema you wish to migrate. The command will automatically convert all custom schema files found in the `slapd.conf` file under the `userat` and or `useroc` headings. However, schema files specified in `INCLUDE` statements are not converted automatically. **This parameter has no effect when the `-i` option is used.**
`-i`	*oldSchemaFile*	Specify the full path and filename of a 4.x schema file to convert for use with Directory Server 5.*x*. **This option will override the `-o` parameter.**
`-s`	*newSchemaFile*	Specify the path and filename of the converted custom schema.
`-x`	*"X-ORIGIN"*	Specify a value for the X-ORIGIN field for attributes and object classes in the converted schema file. Use double quotes ("") to include any space in this value on the command-line. The default string value is "user defined".
`-t`	*traceLevel*	Specify the level of trace, from `0` to `4`, for reporting actions performed during the schema file migration.
`-L`	*logFile*	Specify a file in which to log the migration report.

The Search Operations Audit Tool

The `searchplay` tool will analyze a log file, detect the log entries for search operations, and replay the search operations on the specified directory server. This chapter provides instructions on how to use the `searchplay` tool. It contains the following sections:

- Overview
- Command Usage

Overview

The `searchplay` tool will analyze a log file, detect the log entries for search operations, and replay the search operations on the specified directory server. This functionality is useful for testing the integrity and consistency of directory contents. It may also be used to trace problems that occurred as a result of a sequence of events.

Command Usage

The `searchplay` tool is a simple tool that reads the `access` log file in the current directory and queries the directory server with all of the same search operations in the same order.

Syntax

The syntax of `searchplay` on the command-line takes the following form:

```
searchplay [options] [-f filename] ...
```

Where:

- *options* are the command-line options described in "Options."
- *filename* is an optional parameter to specify a log file other than the current access log. Use this option to specify past log files that have been rotated. The `-f` *filename* option may be repeated to process more than one log file.

Options

The `searchplay` options and parameters are described in Table 27-1.

Table 27-1 Command-Line Options for the `searchplay` Tool

Option	Parameter	Purpose
`-d`		Process the log file in "debug" mode: display the searches found in the log file but do not perform the operations again on the server.
`-s`		Suppress the display of statistics and information about the searches found in the log file. By default, the tool will display which searches it found and will perform again.
`-w`	*seconds*	Specify a time limit, in seconds, for the server to perform each replayed search operation.
`-f`	*filename*	Specify the filename of an alternate log file to replay. You may specify this option more than once to replay several log files. When this option is omitted, `searchplay` will replay the `access` log.
`-h`	*host*	Specify the hostname of an alternate directory server to replay the search requests. Use in conjunction with the `-p` *port* option to compare directory contents and search performance on different directories.
`-p`	*port*	Use in conjunction with `-h` *host* to specify the port of an alternate directory server to replay the search requests.

Chapter 28

The Core File Analyzer Tool

The `viewldbg` tool analyzes the contents of a memory dump file. This chapter provides instructions on how to use the tool. It contains the following sections:

- Overview
- Command Usage
- Command-Line Example

Overview

The `viewldbg` tool reads a core file and extracts information about the runtime execution path and data structures. A *core file* is a copy of the contents of a memory dump, produced when a process is aborted by internal error. The core file must be generated by an executable that was compiled to produce debugging information; it must also be given as input to the `viewldbg` command. It can be used to analyze runtime conditions and errors when developing applications with the Sun™ ONE LDAP SDK for C. The latest version of the tool is in the *DSRK_base*`/bin/dsrk52` directory.

NOTE The `viewldbg` tool is specific to the Solaris™ operating environment and only available on Solaris platform installations.

Command Usage

The `viewldbg` tool is identical to the `viewcore` tool released in version 5.0 of Sun ONE LDAP SDK for C; only the name has changed.

Syntax

The syntax of the `viewldbg` command on the command-line takes the following form:

`viewldbg` *executableFile coreFile outputFile*

Where:

- *executableFile* is the path and filename of the binary executable to debug, the one which produced the core file. This executable must be compiled with debugging options in order for analysis to be possible.
- *coreFile* is the path and filename of the core information from an executable. Core files may be generated automatically when executables crash or generated by the `gcore` command for a running executable. For more information, see the `gcore(1)` man page.
- *outputFile* is path and filename of a file where the `viewldbg` tool will write the result of the core analysis. An example of the output is reproduced in Command-Line Example.

Command-Line Example

The code reproduced in Code Example 28-1 is a sample of the output from the following commands:

```
$ viewldbg executableFile ./core ~/coreOutput.txt
$ cat ~/coreOutput.txt
```

Code Example 28-1 Sample Output for viewldbg

```
Core analysis date. Copyright (c) 2000 Sun Microsystems, Inc.
Currently time
Opening executableFile ./core
architecture is SPARC
0: NOTE offset=0x1234 filesz=0x1234 flags=9 align=8
1: LOAD vaddr=0x0394 filesz=0x0948 memsz=0x0948 mode=RWX
...
```

The Database File Analyzer Tool

The dbscan tool analyzes and extracts information from a Sun™ ONE Directory Server database file. This chapter provides instructions on how to use dbscan. It contains the following sections:

- Overview
- Command Usage

Overview

The dbscan tool analyzes and extracts information from a Directory Server database file. Database files use the .db2 and .db3 extensions in their filename, depending on the version of Directory Server. The tool analyzes these files, extracting information about entries in the Directory Server. The DSRK includes the tool in the *DSRK_base*/bin/dsrk52 directory.

Command Usage

The dbscan tool analyzes *raw* database files and extracts information about the entries stored in the directory. One format of the output generates the LDAP Data Interchange Format (LDIF) representation of directory contents. You can also use this tool to create index files of the database.

Syntax

The syntax of the dbscan tool on the command-line takes the following form:

```
dbscan [ options ] -f filename
```

Where:

- *options* are the command-line options described in "Options."
- *filename* specifies the database filename and path. The tool accepts database files with either .db2 or .db3 extensions.

You will need to redirect the standard output to a file (> *outputFile*) to save it.

Options

The dbscan options and parameters are described in Table 29-1.

Table 29-1 Command-Line Options for dbscan

Option	Parameter	Purpose
-f	*filename*	Specify the database file to analyze.
-i		Analyze the database file to generate its index file in the output.
-e		Analyze the database file to generate a corresponding entry file. The output is in the id2entry format.
-l	*maxLength*	Specify the maximum length of ID lists to display. Longer lists will be truncated at the *maxLength* number of IDs. When this option is not specified, the default maximum length is 4096.
-n		Specify that only ID list lengths be displayed, not their contents.
-G	*minSize*	Specify that only index entries with more than *minSize* IDs should be displayed in the output. **This option is only effective when used with the -n option.**
-r		Show libdb record numbers in the output.
-k	*key*	Look for and output only the specified *key* in the database.

Chapter 30

Network Security Services

The Network Security Services are a set of open source libraries and tools used to implement and deploy applications for Internet security based on open standards. The security tools help to perform diagnostics, manage certificates, keys and cryptography modules, and debug SSL- and TLS-based applications. This chapter provides information on the tools. It contains the following sections:

- Overview
- Security Standards
- Managing Certificates and Keys
- Other Utilities

Overview

Network Security Services (NSS), which is built on the same code base as the original Netscape Security Services, is available from `mozilla.org` as open source software. The NSS tools are introduced in this chapter. Each tool is briefly described, and followed by a URL from which you can access more complete documentation. The DSRK includes the NSS in the *DSRK_base*`/lib/nss/` directory.

NOTE If you are unfamiliar with the concepts of Internet security, you should start with this overview at:

`http://www.mozilla.org/projects/security/pki/nss/overview.html`

In addition to libraries and APIs, NSS provides security tools required for debugging, diagnostics, certificate and key management, cryptography module management, and other development tasks. The security tools are located in the *DSRK_base*/lib/nss/bin directory. In order to run them, you will need to add the following library locations to your LD_LIBRARY_PATH environment variable:

DSRK_base/lib/nss/lib:*DSRK_base*/lib

Security Standards

This section describes the security standards implemented by the NSS libraries and tools. More information and links to the official standards documents may be found on the Mozilla web site.

SSL v2 and v3 The Secure Sockets Layer (SSL) protocol allows mutual authentication between a client and server and the establishment of an authenticated and encrypted connection.

TLS v1 (RFC 2246) The Transport Layer Security (TLS) protocol from the IETF (Internet Engineering Task Force) will eventually supersede SSL while remaining backward-compatible with SSL implementations.

PKCS (Public-Key Cryptography System) #1 PKCS #1 is an RSA standard that governs implementation of public-key cryptography based on the RSA algorithm (invented by Rivest, Shamir, Aldeman). This algorithm is the basis for all other PKCS standards.

PKCS #3 PKCS #3 is an RSA standard that governs implementation of the Diffie-Hellman key agreement.

PKCS #5 PKCS #5 is an RSA standard that governs password-based cryptography. For example, it can be used to encrypt private keys for storage.

PKCS #7 PKCS #7 is an RSA standard that governs the application of cryptography to data, such as for digital signatures.

PKCS #8 PKCS #8 is an RSA standard that governs the storage and encryption of private keys.

PKCS #9 PKCS #9 is an RSA standard that governs selected attribute types, including those used with PKCS #7, PKCS #8, and PKCS #10.

PKCS #10 PKCS #10 is an RSA standard that governs the syntax for certificate requests.

PKCS #11 PKCS #11 is an RSA standard that governs communication with cryptographic tokens (such as hardware accelerators and smart cards) and permits application independence from specific algorithms and implementations.

PKCS #12 PKCS #12 is an RSA standard that governs the format used to store or transport private keys, certificates, and other secret material.

S/MIME (RFC 2311 and RFC 2633) S/MIME (Secure/Multipurpose Internet Mail Extensions) is an IETF message specification based on the popular MIME standard. It provides a consistent way to send and receive signed and encrypted MIME data.

X.509 v3 This is an International Telecommunication Union (ITU) standard that governs the format of certificates used for authentication in public-key cryptography.

OCSP (RFC 2560) The Online Certificate Status Protocol (OCSP) governs real-time confirmation of certificate validity.

PKIX Certificate and CRL Profile (RFC 2459) The first part of the four-part standard under development by the Public-Key Infrastructure (X.509) working group of the IETF (known as PKIX) for a public-key infrastructure for the Internet.

AES, RSA, DSA, Triple DES, DES, Diffie-Hellman, RC2, RC4, SHA-1, MD2, MD5 These are common cryptographic algorithms used in public-key and symmetric-key cryptography.

Managing Certificates and Keys

The tools described in this section store, retrieve and protect the keys and certificates on which encryption and identification rely. *Understanding Certificates*, a chapter from the documentation for Sun Microsystem's discontinued Certificate Management System, provides information on how keys and certificates are used to protect data and identity in secure communication.

certutil

The Certificate Database Tool, `certutil`, is a command-line utility that can create and modify the `cert7.db` and `key3.db` database files. The key and certificate management process generally begins with creating keys in the key database, then generating and managing certificates in the certificate database. More information on `certutil` can be found at:

www.mozilla.org/projects/security/pki/nss/tools/certutil.html

cmsutil

The cmsutil command-line utility uses the S/MIME Toolkit to perform basic operations, such as encryption and decryption, on Cryptographic Message Syntax (CMS) messages. It performs basic certificate management operations such as encrypting, decrypting, and signing messages. More information on cmsutil can be found at:

www.mozilla.org/projects/security/pki/nss/tools/cmsutil.html

modutil

The Security Module Database Tool, modutil, is a command-line utility for managing the database of PKCS #11 modules (secmod.db files). You can use the tool to add and delete PKCS #11 modules, change passwords, set defaults, list module contents, enable or disable slots, enable or disable FIPS-140-1 compliance, and assign default providers for cryptographic operations. More information on modutil can be found at:

www.mozilla.org/projects/security/pki/nss/tools/modutil.html

pk12util

The pk12util command-line utility imports and exports both keys and certificates, defined by the PKCS #12 standard, to and from their respective database and file formats. More information on pk12util can be found at:

www.mozilla.org/projects/security/pki/nss/tools/pk12util.html

signtool

The signtool command creates signed jar archives containing files, code, or both. This command is fully documented in the documentation for Sun Microsystem's discontinued Certificate Management System. This can be found at:

http://docs.sun.com/source/816-5543-10/index.html

signver

The `signver` tool verifies signatures on digitally-signed objects, such as jar files signed with `signtool`. This command is fully documented in the documentation for Sun Microsystem's discontinued Certificate Management System. This can be found at:

```
http://docs.sun.com/source/816-5543-10/index.html
```

ssltap

The SSL Debugging Tool, `ssltap`, is an SSL-aware command-line proxy. It can proxy requests for an SSL server and display the contents of the messages exchanged between the client and server. It watches TCP connections and displays the data going by. If a connection is SSL, the data display includes interpreted SSL records and handshaking. More information can be found at either of the following URLs:

```
www.mozilla.org/projects/security/pki/nss/tools/ssltap.html
http://docs.sun.com/source/816-5549-10/ssltap.htm
```

Format Conversion Commands

The following are format conversion commands. More information can be found in the Sun ONE Certificate Server 4.7 Command-Line Tools Guide and by invoking the command without any arguments on the command-line.

atob

`atob` converts ASCII base-64 encoded data to binary base-64 encoded data. It reads an encoded file, strips off any leading and trailing lines added by mailers, and recreates a copy of the original file on the standard output.

btoa

`btoa` converts binary base-64 encoded data to ASCII base-64 encoded data. It is a filter that reads anything from standard input, and encodes it into printable ASCII on the standard output. It also attaches a header and checksum information used by the reverse filter `atob` to find the start of the data and to check integrity.

Other Utilities

This section lists the remaining commands provided in the *DSRK_base*/lib/nss/bin directory. Minimal documentation detailing each tool's options is available by invoking the command without any arguments. The tools are:

- bltest
- makepqg
- pp
- certcgi
- newuser
- rsaperf
- checkcert
- ocspclnt
- sdrtest
- client
- oidcalc
- selfserv
- crlutil
- p7content
- strsclnt
- derdump
- p7env
- tstclnt
- digest
- p7sign
- instinit
- p7verify

Chapter 31

Unsupported Utilities

This chapter provides brief descriptions of the unsupported utilities installed with the Sun™ ONE Directory Server Resource Kit (DSRK). It contains the following sections:

- PerLDAP
- Perl Scripts

PerLDAP

PerLDAP is a third-party tool that has libraries of Perl routines for accessing a directory server and performing LDAP operations. The source code and more information about it can be found at:

```
http://www.mozilla.org/directory/perldap.html
```

In order to use the PerLDAP libraries installed along with the Directory Server Resource Kit, you must first build and install its libraries. In this procedure, *DSRK_base* refers to the base directory of your Directory Server Resource Kit installation. By default this is `/opt/iPlanet`. The procedure is also described in the file *DSRK_base*`/unsupported/perLdap/INSTALL`.

1. Change to the directory in which PerLDAP was installed.

   ```
   % cd DSRK_base/unsupported/perLdap
   ```

2. Run the Perl makefile.

   ```
   % perl Makefile.PL PREFIX=DSRK_base LIB=DSRK_base
   ```

3. Enter the information in bold, when prompted.

Code Example 31-1 Information Needed for perLDAP Build

```
PerLDAP - Perl 5 Module for LDAP
================================
Directory containing 'include' and 'lib' directory of the iPlanet
LDAP Software Developer Kit (default: /usr): DSRK_base/lib/ldapcsdk
Using LDAPv3 Developer Kit (default: yes)?  yes
Include SSL Support (default: yes)?  yes
Located multiple libraries:
  - libldap50.so
  - libssldap50.so
  - libprldap50.so
Libraries to link with(default:-LinstallDir/lib/ldapcsdk/lib -lldap50):
-LDSRK_base/lib/ldapcsdk/lib -lssldap50 -lssl3 -lprldap50 -lplds4
-lplc4 -lnss3 -lnspr4 -lldap50 -lldif50 -llber50 -liutil50
Writing Makefile for Mozilla::LDAP::API
```

4. Install the perLDAP libraries with the make utility:

```
% make install
```

You can now run any Perl script requiring the PerLDAP libraries as follows:

```
% perl -IDSRK_base script.pl
```

Perl Scripts

The unsupported Perl scripts are small utilities that perform a variety of maintenance tasks. Some perform specific, programmed actions and will not work outside of their designated context. Others do not perform error checking and should be used only if you understand their behavior.

CAUTION Sun Microsystems makes no claim as to the suitability or correctness of these scripts. Use them at your own risk.

The scripts discussed in Table 31-1 on page 317 are provided solely as examples of Perl shell programming and examples of using PerLDAP routines to access a directory.

Table 31-1 Unsupported Utilities in *DSRK_base*/unsupported/perl

Script Name	Intended Usage
adduser.pl	Add users to an LDAP directory.
changes2ldif.pl	Find an entry with a given changenumber.
corescoop.pl	Move mail server core files into a publishable area.
dn2ldif.pl	Read a set of DNs and produce the LDIF output.
export_auto_home.pl	Generate the equivalent of a /etc/auto.home file with data extracted from an LDAP directory.
export_mailgroups.pl	Generate the equivalent of a /etc/mailgroups.aliases file with data extracted from an from LDAP directory.
fixcopiedfrom.pl	Modify the copiedfrom attribute, which is sometimes needed to get replication working.
genaliases.pl	Generate the equivalent of a /etc/mlm-aliases/aliases.{intern,extern} file with data extracted from an LDAP directory.
genpasswd.pl	Generate the equivalent of a /etc/passwd file with data extracted from an LDAP directory.
import_aka.pl	Import the information of a SmartList mail alias into an LDAP directory.
import_auto_home.pl	Import the NIS auto.home map into an LDAP directory.
import_epage.pl	Import the EtherPage epage.users file into an LDAP directory.
inconsist.pl	Reports inconsistencies between LDAP and NIS data.
layoff.pl	Gives an example of how to automate certain system administration tasks.
ldap_mail.pl	Update mailRecipient information for a user.
ldap_migrate.pl	Move an LDAP entry from one specific server to another.
ldap_stress.pl	Perform a stress test with searches simulating Message Server load on an LDAP server.
ldappasswd.pl	Change the password of one or more users.
ldapstats.pl	Gather statistics from an LDAP server.
lfinger.pl	Perform an LDAP search with a command-line that emulates the UNIX finger command.
mgroup.pl	Manage a mailing list or any other group from the command-line.
migrate_user.pl	Move an entry from one subtree to another by creating the new entry, copying the attributes, and deleting the old entry.

Table 31-1 Unsupported Utilities in *DSRK_base*/unsupported/perl *(Continued)*

Script Name	Intended Usage
modclass.pl	Add or delete one or more object classes from one or more entries.
normphones.pl	Normalize phone numbers and make sure they have the correct US area code.
qsearch.pl	Perform a quick and simple LDAP search for an entry.
rand_mods.pl	Modify or delete an attribute for one or more entries.
renattr.pl	Rename one or more attributes in entries matching the search criteria.
repstat.pl	Check the replication status on a Master and its replica.
restarter.pl	Restart a DS server under certain conditions.
rmduplicates.pl	Remove duplicate attribute values from all entries.
rmentry.pl	Remove one or more LDAP entries interactively.
tabdump.pl	Generate a tab-separated output of entries matching the search criteria.
termuser.pl	Similar to layoff.pl: gives an example of how to automate certain system administration tasks.
uidsynch.pl	Synchronize all UIDs with their mail attribute.
vrfyPO.pl	Check the consistency between the data on a mail server and in an LDAP directory.
vrfymail.pl	Verify that the mail and mailalternate attributes are not duplicates for any entry in a directory.
wits_dump.pl	Display a subset of attribute values for all entries in the directory.

Part 7

Sample Phonebook Application

Chapter 32

JSP Directory Gateway Phonebook

The JavaServer Pages™ Directory Gateway is a sample application that demonstrates how a web-based client may access Sun™ ONE Directory Server through JSP technology. The application presents a simple browser interface for searching and editing the contents of an enterprise phonebook. This chapter provides instructions on how to use the application. It contains the following sections:

- Overview
- Software Installation
- Post-Installation Configuration
- Accessing LookMeUp

Overview

A *gateway* is a client that lives on a HyperText Transfer Protocol (HTTP) server and offers access to Lightweight Directory Access Protocol (LDAP) directory data using a web browser. A gateway can be used to perform directory lookup, or to authenticate to the directory and complete database administration tasks. The JavaServer Pages (JSP) Directory Gateway (jdgw) is a phonebook application that can be used to access any directory containing entries representing people. The phonebook application itself is called LookMeUp. It is comprised of three components that allow an LDAP directory to be accessible through HTML. All three components can be found in the *DSRK_base*/`jdgw` directory. They are:

- Apache Software
- LDAP Tag Libraries
- LookMeUp Application Files

Apache Software

The *DSRK_base*/jdgw/apache-downloads directory itself contains three products from the Jakarta project of The Apache Software Foundation (ASF).

NOTE The Jakarta Project creates and maintains open source Java solutions.

Included with the DSRK are the following Jakarta components.

- Apache Ant 1.4.1 is a Java-based build tool. It is in the *DSRK_base*/jdgw/apache-downloads/ant1.4.1 directory.
- The JavaServer Pages Tag Libraries (JSPTL) standard can be found in the *DSRK_base*/jdgw/apache-downloads/jsptl1.2 directory. The tags are XML-like structures that contain the logic to generate the HTML page contents. Chapter 33, "Tag Library Reference" details the tag libraries.
- The Tomcat web server provides a framework from which LookMeUp's JSP can be run.

LDAP Tag Libraries

The *DSRK_base*/jdgw/ldap-taglib directory contains a LDAP tag library which uses the Sun ONE LDAP SDK for Java to query a directory server and process the response. You may use the LDAP tag library to develop new JSP to implement your own version of the gateway application. Again, Chapter 33, "Tag Library Reference" details the tag libraries.

LookMeUp Application Files

The *DSRK_base*/jdgw/phonebook-app directory contains the set of JSP and related files needed to set up the phonebook application called LookMeUp. You can customize the JSP files to tailor the HTML output.

Software Installation

The installation procedure for the JSP Directory Gateway is:

1. Download and install the Java 2 Platform, Standard Edition, version 1.3.1 if you do not already have it. More information can be found in "Java 2 Platform, v1.3.1."
2. Install the Tomcat web server provided with the DSRK, or download and install the latest version. More information can be found in "Tomcat Web Server, v4.0 or Later."
3. Install version 1.2 of the JavaServer Pages Tag Library (JSPTL) provided with the DSRK. More information can be found in "JSP Standard Tag Library."
4. Configure LookMeUp to access your LDAP directory server and the web server to use the `jdgw` files. More information can be found in "Post-Installation Configuration."

Java 2 Platform, v1.3.1

The JSPTL included with LookMeUp requires the Java 2 Platform, Standard Edition (J2SE™), version 1.3.1, also know as the JDK™ 1.3.1. If you do not already have this software, it is available from Sun Microsystems at:

```
http://java.sun.com/j2se/1.3/
```

Download the software and run the setup program, following the installation instructions provided with the download.

NOTE The instructions in this chapter will use *JDK_base* to refer to the directory where you install J2SE 1.3.1.

On UNIX® platforms, you will need to add *JDK_base*`/bin` to your environment's PATH variable. On Windows® platforms, the installer will automatically update your registry with the JDK path.

Tomcat Web Server, v4.0 or Later

The DSRK includes files for installing the Tomcat web server, version 4.01. Use one of the files in the *DSRK_base*`/jdgw/apache-downloads/tomcat4.01/` directory, according to your platform:

- `jakarta-tomcat-4.0.1.tar.gz` or `jakarta-tomcat-4.0.1.zip` on Solaris and UNIX platforms. These files contain all Tomcat web server files, which you should uncompress in the directory where you wish to install the product using:
  - `gzip -d < jakarta-tomcat-4.0.1.tar.gz | tar xvfm -` for the `.gz` file.
  - `unzip jakarta-tomcat-4.0.1.zip` for the `.zip` file.
- `jakarta-tomcat-4.0.1.exe` is a self-extracting installer program for Windows platforms. Run this file and choose a location for the installation.

Alternately, you may download a later version from the Jakarta project web site and install it in the same manner:

`http://jakarta.apache.org/site/binindex.html`

NOTE The instructions in this chapter will use *Tomcat_base* to refer to the directory where you install the Tomcat web server. Catalina is the name of the Tomcat 4.0 servlet container.

Starting the Server

Once installed, start the web server with one of the following sets of commands, according to your platform:

- On a Solaris™ or UNIX platform, using the Korn shell for example:

  ```
  export CATALINA_HOME=Tomcat_base
  export JAVA_HOME=JDK_base
  Tomcat_base/bin/startup.sh
  ```

- On a Windows platform, in a DOS command window:

  ```
  set CATALINA_HOME=Tomcat_base
  set JAVA_HOME=JDK_base
  Tomcat_base\bin\startup.bat
  ```

The file `RUNNING.txt` (found in the directory in which the application is uncompressed) includes more information.

Testing the Server and Accessing Tomcat Documentation

Test your Tomcat web server installation by accessing the following URL in any browser:

`http://localhost:8080/ or http://hostname:8080/`

You should see the default `index.html` file of the server. From this page, you can access the links to run examples of JSP pages and servlets. There is also a link to the Tomcat documentation which is installed locally:

```
http://localhost:8080/tomcat-docs/index.html
```

JSP Standard Tag Library

The DSRK installation includes the files for installing the JSP Tag Library (JSPTL), Early Access Release 1.2, which was the latest available at release time. Installing the JSPTL is optional because LookMeUp already contains a copy of the tag library's `jar` file but, if you wish to learn about the tag library by viewing its documentation and running its examples, the full JSPTL needs to be installed. (Chapter 33, "Tag Library Reference" details the libraries themselves.) Use one of the following files in the *DSRK_base*`/jdgw/apache-downloads/jsptl1.2/` directory to install:

- `jakarta-taglibs-jsptl-ea1.2.tar.gz` on Solaris and UNIX platforms. Uncompress this file in the directory of your choice.
- `jakarta-taglibs-jsptl-ea1.2.zip` on Windows platforms. Uncompress this file in the directory of your choice.

NOTE Later versions of the included JSPTL are now abbreviated and known as JavaServer Pages Standard Tag Library (JSTL). The new versions have a different structure and its files are not compatible with the `jdgw` application. You may still download the new release to view its documentation and run the examples, but the application must use the older JSPTL `jar` files.

Viewing the JSPTL Documentation

To view the JSPTL documentation and examples, you must edit the configuration file named *Tomcat_base*`/conf/server.xml` to include the definitions below the root context definition reproduced in Code Example 32-1. These definitions assume that the JSPTL is installed in a directory named *JSPTL_base*.

Code Example 32-1 server.xml Configuration File

```
<!-- JSPTL Doc Context -->
<Context path="/jsptl-doc" docBase="JSPTL_base/jsptl-doc.war"
         debug="0" reloadable="true">
    <Logger className="org.apache.catalina.logger.FileLogger"
            prefix="jsptl_doc_log." suffix=".txt"
            timestamp="true"/>
</Context>
```

Code Example 32-1 server.xml Configuration File *(Continued)*

```
<!-- JSPTL Examples Context -->
<Context path="/jsptl-examples" debug="0" reloadable="true"
         docBase="JSPTL_base/jsptl-examples.war">
    <Logger className="org.apache.catalina.logger.FileLogger"
            prefix="jsptl_examples_log." suffix=".txt"
            timestamp="true"/>
</Context>
```

After modifying the server configuration file, restart the Tomcat server with the following commands:

- On a Solaris or UNIX platform:

  Tomcat_base/bin/shutdown.sh
  Tomcat_base/bin/startup.sh

- On a Windows platform, in the same DOS command window:

  Tomcat_base\bin\shutdown.bat
  Tomcat_base\bin\startup.bat

The JSPTL documentation and examples should now be available at:

```
http://localhost:8080/jsptl-doc
http://localhost:8080/jsptl-examples
```

Post-Installation Configuration

Once the required software is installed, you will need to configure the LookMeUp application and the Tomcat server.

LookMeUp Configuration

LookMeUp uses the configuration files located in the *DSRK_base*/jdgw/phonebook-app/WEB-INF/classes directory. You need to edit the file named lookmeup.properties to specify the directory server that will be queried by LookMeUp.

CAUTION The `screen.properties` file and the `ldaperrors.properties` file contain text that is used by the JSP files to generate HTML output. You do not need to modify these files unless you wish to change the output of the sample application.

Code Example 32-2 is an example of `lookmeup.properties` (which also contains parameters for specifying the base or suffix of all phone book searches and the country and language locale of input and output text).

Code Example 32-2 lookmeup.properties File

```
hostname=directory.example.com
port=389
base=ou=people,dc=example,dc=com
country=US
language=en
```

Tomcat Server Configuration

You also need to add the context definition for LookMeUp to the Tomcat server configuration file `server.xml` located in the *Tomcat_base*`/conf/` directory. `server.xml` is duplicated in Code Example 32-3.

Code Example 32-3 server.xml File

```
<!-- JDGW LookMeUp phone book application -->
<Context path="/jdgw" docBase="InstallDir/jdgw/phonebook-app"
         debug="0" reloadable="true">
    <Logger className="org.apache.catalina.logger.FileLogger"
            prefix="jdgw_log." suffix=".txt"
            timestamp="true"/>
</Context>
```

After modifying `server.xml`, restart the Tomcat server using either of the following:

- On a Solaris or UNIX platform:

  Tomcat_base`/bin/shutdown.sh`
  Tomcat_base`/bin/startup.sh`

- On a Windows platform, in the same DOS command window:

Tomcat_base\bin\shutdown.bat
Tomcat_base\bin\startup.bat

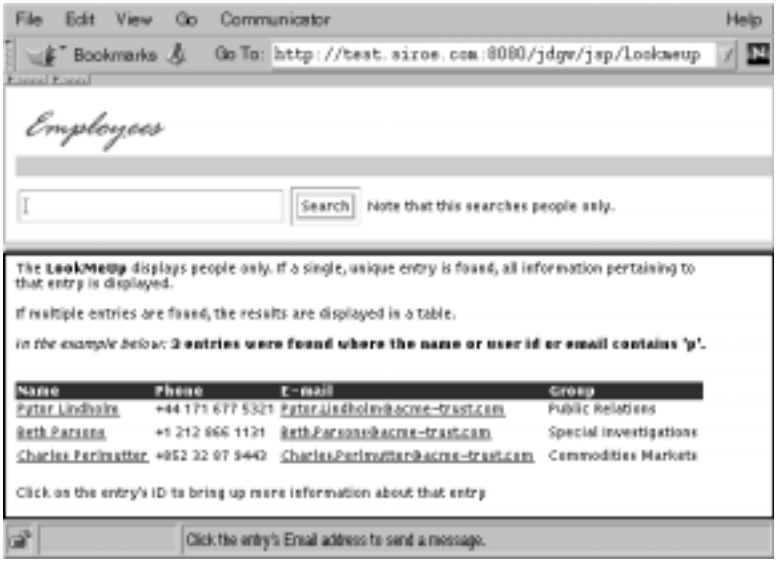
Accessing LookMeUp

When the full installation and file configuration are successfully completed, you will be able to access LookMeUp at:

```
http://localhost:8080/jdgw/jsp/lookmeup
```

LookMeUp is a simple employee phonebook. It will search for people in your directory and display the results in easy-to-read tables. The basic elements of the application are templates that contain HTML elements with LDAPTL, JSPTL, and UTILTL elements in places where dynamic content will be inserted in the generated page sent to the client browser. Figure 32-1 shows the introductory search page for the application.

Figure 32-1 Index Page of the LookMeUp Application



Chapter 33

Tag Library Reference

This chapter provides a detailed reference of the Lightweight Directory Access Protocol (LDAP) Tag Libraries, the Utilities Tag Libraries and the JavaServer Pages™ Standard Tag Library (JSPTL) included with the Sun™ ONE Directory Server Resource Kit (DSRK). This information can help you customize the phone book application (called LookMeUp) or develop other web-based LDAP clients. It contains the following sections:

- Overview
- LDAP Tag Library
- Utilities Tag Library
- JSP Standard Tag Library

Overview

JavaServer Pages (JSP) allow developers to dynamically generate HTML, XML or other types of web pages. The technology allows Java code to be embedded into static content. The code will most probably include a set of pre-defined *actions* (standardized in the JSP specification) to handle the core functionality common to most JSP, including database access, and XML processing. JSP also supports the development of reusable modules to handle other functions; these are referred to as *custom actions*. All actions are invoked by *tags* defined using an XML syntax (a start tag, an end tag, and possibly a body). Code Example 33-1 on page 330 is a simple HTML page using a pre-defined action (`include`) surrounded by a tag. (`jsp:` is the start tag and, because there is no body, `/>` is the end tag.)

Code Example 33-1 Simple Action and Tag

```
<HTML>
<BODY>
Going to include hello.jsp...<BR>
<jsp:include page="hello.jsp"/>
</BODY>
</HTML>
```

A *tag library* is a collection of tags (and their corresponding actions). Tags are grouped in libraries according to their function. The LookMeUp application relies on the LDAP Tag Library to perform LDAP queries, the Utilities Tag Library to form said queries and process responses, and the JSP Standard Tag Library, Early Access Release 1.2, for iterations and conditional operations.

LDAP Tag Library

The LDAP Tag Library (LDAPTL) implements the interface between a JSP-enabled web server and LDAP v3 directories. It sits in top of the JSP Standard Tag Library which performs LDAP operations on the directory server. The LDAPTL represents LDAP attributes, entries, and operations, as well as the results of operations. The implementation of these tags uses the Sun ONE LDAP SDK for Java to interact with any directory server. The tag library definition is contained in the `ldaptl.tld` found in the *DSRK_base*`/jdgw/phonebook-app/WEB-INF` directory.

Summary

The following material is relevant when using the LDAPTL.

- All tags for performing LDAP operations have `host` and `port` attributes to specify the directory server. Both attributes are optional and, when omitted, LDAP operations will use the standard port 389 on the local host.
- The `user` and `password` attributes are used to bind to the directory server and are also optional. If not provided, an anonymous bind will be performed.

- The <ldap:add> and <ldap:modify> tags are the only ones in the library to contain other tags. The add operation defines the new entry as a set of associated <ldap:attr> tags that may be either contained in the <ldap:add> tag or referenced in a variable. In a similar manner, the <ldap:modify> tag may either contain or reference <ldap:mod> tags to define individual modifications.
- The LDAP operation tags will return LDAP exceptions in the variable specified in their response attribute. Check this variable to determine if an error has occurred while processing the operation. Explanations for errors may be displayed by accessing the text in the ldaperrors.properties file through a <util:prop> tag.

<ldap:add>

Add a set of attribute-value pairs as a new entry in the directory.

DESCRIPTION

The <ldap:add> element either references or contains a set of attribute-value pairs that will become the contents of the new entry. Attributes and their values are defined by the <ldap:attr> element. You must define all required attributes for the new entry, including any naming attributes and objectclass attributes. If schema checking is enabled in your directory, the other attributes and their values must conform to any defined object class.

SYNTAX

```
<ldap:add [host="hostname" port="port"]
          user="bindDN" passwd="password"
          dn="newDN" attrs="varName" [response="varName"]/>

<ldap:add [host="hostname" port="port"]
          user="bindDN" passwd="password"
          dn="newDN" [response="varName"]>
    <ldap:attr … /> …
</ldap:add>
```

ATTRIBUTES

host="*hostname*"	Specify the hostname of the directory server. When this attribute is omitted, the default is localhost.
port="*port*"	Specify the port number of the directory server on the given host. When this attribute is omitted, the default port is 389.
user="*bindDN*"	Specify a bind DN for accessing your directory. You must specify a bind DN that has write permission in the new DN's suffix.

passwd="*password*"	Specify the password for the bind DN. **CAUTION: the password is visible in this tag, so the JSP file should be read-protected.**
dn="*newDN*"	Specify the full DN of the new entry.
attrs="*varName*"	Give the name of a variable containing the set of attribute-value definitions for the new entry. When this attribute is omitted, the <ldap:add> tag must contain the <ldap:attr> definitions.
response="*varName*"	Give the name of a variable (in the PAGE scope) for storing any error response. When omitted, the default name is response. If an LDAP error occurs during the operation, this variable will be set to the corresponding LDAPException Bean.

EXAMPLES

1. The following example shows attribute definitions referenced by the attrs attribute of the <ldap:add> element:

```
<ldap:attr var="attrs" scope="request"
            name="objectclass" value="top"/>
<ldap:attr var="attrs" scope="request"
            name="objectclass" value="organizationalunit"/>
<ldap:attr var="attrs" scope="request"
            name="ou" value="People"/>
<ldap:attr var="attrs" scope="request"
            name="description" value="Employee branch"/>

<ldap:add host="phonebook.example.com"
           user="cn=Directory Manager" passwd="password"
           dn="ou=People,dc=example,dc=com"
           attrs="$request:attrs" response="addError"/>
```

2. The second example creates an identical entry, but the attributes are defined within the <ldap:add> tag:

```
<ldap:add host="phonebook.example.com"
           user="cn=Directory Manager" passwd="password"
           dn="ou=People,dc=example,dc=com" response="addError">
   <ldap:attr name="objectclass" value="top"/>
   <ldap:attr name="objectclass" value="organizationalunit"/>
   <ldap:attr name="ou" value="People"/>
   <ldap:attr name="description" value="Employee branch"/>
</ldap:add>
```

SEE ALSO

<ldap:attr>

<ldap:attr>

Defines an attribute-value pair for the <ldap:add> element or retrieves an attribute-value pair from an <ldap:search> result.

DESCRIPTION

The <ldap:attr> element represents an LDAP attribute, its name and its values. It is used with the <ldap:add> element to define the attributes of a new entry. It also simplifies the access to attributes and their values in <ldap:search> results.

This element may appear inside the <ldap:add> element where it defines the name and value of a new attribute. It may also be referenced from outside the <ldap:add> element, in which case it must also name a variable and optional scope where the definition is stored. See <ldap:search> for more information.

When processing search results, the <ldap:attr> element extracts the value or values of the named attribute for display or storage in a variable. Use the entry attribute to name the variable containing search results, either inside or outside of an <ldap:search> element.

SYNTAX

For defining an attribute:

```
<ldap:attr name="attrName" value="valueString"
            [ var="varName" scope="scopeName" ]/>
```

For displaying an attribute:

```
<ldap:attr name="attrName" entry="varName"
            [ default="valueString" op="value|values" ]
            [ var="varName" scope="scopeName" ]/>
```

ATTRIBUTES

name="*attrName*"	Give the name of the LDAP attribute to define or retrieve.
value="*valueString*"	Specify the value when defining an attribute to add.
var="*varName*"	Give the name of the variable where this attribute definition will be stored. The variable name and scope may also be used to store the retrieved value for later access, such as for iterating through a multi-valued attribute. (The variable name and scope are not required when the <ldap:attr> tag is contained in the <ldap:add> element.)
scope="*varScope*"	Give the scope of the variable where this attribute definition or values will be stored.
entry="*varName*"	Specify the name of variable that contains an entry of the search results.

default="*valueString*"	Specify a default value to store or display when the attribute is missing or when its value is null or empty.
op="*operation*"	Specify the extent of the value to retrieve: • value - Retrieve a single value or the first of a multi-valued attribute; this is the default behavior. • values - Retrieve all attribute values as an array.

SEE ALSO

<ldap:add> and its examples, <ldap:search> and its examples.

<ldap:delete>

Delete an entry from the directory.

DESCRIPTION

Evaluating this tag will perform the LDAP delete operation on the given DN. Check the variable given by the response attribute to determine if the operation was successful.

SYNTAX

```
<ldap:delete [host="hostname" port="port" ]
             user="bindDN" passwd="password"
             dn="oldDN" response="varName"/>
```

ATTRIBUTES

host="*hostname*"	Specify the hostname of the directory server. When this attribute is omitted, the default is localhost.
port="*port*"	Specify the port number of the directory server on the given host. When this attribute is omitted, the default port is 389.
user="*bindDN*"	Specify a bind DN for accessing your directory. You must specify a bind DN that has permission to remove the given entry.
passwd="*password*"	Specify the password for the bind DN. **CAUTION: the password is visible in this tag, so the JSP file should be read-protected.**
dn="*oldDN*"	Specify the full DN of the entry to delete.
response="*varName*"	Give the name of a variable (in the PAGE scope) for storing any error response. When omitted, the default name is response. If an LDAP error occurs during the operation, this variable will be set to the corresponding LDAPException Bean.

EXAMPLE

```
<ldap:delete host="phonebook.example.com"
             user="cn=Directory Manager" passwd="password"
             dn="uid=007,ou=People,dc=example,dc=com"
             response="deleteError"/>
```

<ldap:dn>

Normalize or extract distinguished name (DN) components for display or storage.

DESCRIPTION

This element extracts the comma-separated components of a distinguished name (DN), according to the op attribute. Specify a variable name and optional scope to store the component string or array of strings in a variable. If none is specified, the element will display component in the output. If no operation is specified, the element will normalize the DN string. The DN is specified either as the value attribute or as the tag's contents.

SYNTAX

```
<ldap:dn value="DNstring" [ op="leaf|leaf-value|parts|value-parts" ]
         [ var="varName" scope="scopeName" ]/>

<ldap:dn [ op="leaf|leaf-value|parts|value-parts" ]
         [ var="varName" scope="scopeName" ]>
    DNstring
</ldap:dn>
```

ATTRIBUTES

`value="`*DNstring*`"`	Specify the DN on which to operate. This attribute is optional because the DN may also be specified as the element's contents.
`op="`*operation*`"`	Specify the DN component or components to extract from the DN: • `leaf` - Extracts the DN's first naming attribute and value. • `leaf-value` - Extracts only the first naming value from the DN. • `parts` - Returns an array of each of the DN's components. • `value-parts` - Returns an array containing only the DN's values.
`var="`*varName*`"`	Give the name of the variable where the tag's output will be stored.
`scope="`*varScope*`"`	Give the scope of the variable where the tag's output will be stored.

EXAMPLES

```
INPUT DN STRING: 'cn=Babs Jensen,ou=People,dc=example,dc=com'

Normalize =
    <ldap:dn>cn=Babs Jensen,ou=People,dc=example,dc=com</ldap:dn><br>

Leaf Value =
    <ldap:dn value="cn=Babs Jensen,ou=People,dc=example,dc=com"
             op="leaf-value"/><br>

Leaf =
    <ldap:dn value="cn=Babs Jensen,ou=People,dc=example,dc=com"
             op="leaf"/><br>

<ldap:dn value="cn=Babs Jensen,ou=People,dc=example,dc=com"
         op="value-parts" var="values"/>
<jx:forEach var="val" items="$valuess">
    Value Part: <jx:expr value="$val"/><br>
</jx:forEach>

<ldap:dn value="cn=Babs Jensen,ou=People,dc=example,dc=com"
         op="parts" var="parts"/>
<jx:forEach var="part" items="$parts">
    Part: <jx:expr value="$part"/><br>
</jx:forEach>
```

<ldap:mod>

Define a modification to a single attribute. One or more instances of this element are either contained in or referenced by the <ldap:modify> element.

DESCRIPTION

Each instance of this element defines a modification to perform on a single attribute of an entry. A modification may perform one of the following changes:

- Add a new attribute and its value to the entry.
- Replace an existing attribute's value.
- Delete one value of a multivalued attribute.
- Delete all values of an attribute, effectively removing it from the entry.

Use multiple instances to create a set of modifications for the <ldap:modify> tag to perform. The <ldap:modify> tag must either contain <ldap:mod> elements or reference the variable named by the var and scope attributes.

SYNTAX

```
<ldap:mod op="add|delete|replace"
          name="attrName" [ value="valueString" ]
          [ var="varName" scope="scopeName" ]>
```

ATTRIBUTES

op="*operation*"	Specify the type of modification to perform on the given attribute- value pair. The *operation* is either add, delete, or replace. When omitted, the default operation is to replace the attribute value.
name="*attrName*"	Give the name of the attribute to be modified.
value="*valueString*"	Give the new value of the attribute to be added or replaced. When deleting an attribute, either specify a particular value to remove from a multivalued attribute, or do not specify a value to remove all values and delete the attribute from the entry.
var="*varName*"	Give the name of the variable where this modification definition will be stored. The variable name and scope are not required when the <ldap:mod> tag is contained in the <ldap:modify> element.
scope="*varScope*"	Give the scope of the variable where this modification definition will be stored. When omitted, the PAGE scope is the default.

SEE ALSO

<ldap:modify> and its examples.

<ldap:modify>

Perform a set of modifications on attributes of a single entry.

DESCRIPTION

The <ldap:modify> element either references or contains a set of attribute modifications that will be performed on the target entry. Attributes may be added, replaced, or deleted, as defined in the <ldap:mod> element.

You must provide the authentication for a bind DN that has write permission on the target DN. If schema checking is enabled in your directory, new attributes or new values must conform to the object class of the target DN. Write permission or schema errors will cause the operation to fail, which you can detect in the variable named by the response attribute.

SYNTAX

```
<ldap:modify [host="hostname" port="port" ]
              user="bindDN" passwd="password"
              dn="targetDN" mods="varName" response="varName"/>

<ldap:modify [host="hostname" port="port" ]
              user="bindDN" passwd="password"
              dn="targetDN" response="varName">
    <ldap:mod … /> …
</ldap:modify>

<ldap:modify >
```

ATTRIBUTES

`host="`*hostname*`"`	Specify the hostname of the directory server. When this attribute is omitted, the default is `localhost`.
`port="`*port*`"`	Specify the port number of the directory server on the given host. When this attribute is omitted, the default port is 389.
`user="`*bindDN*`"`	Specify a bind DN for accessing your directory. You must specify a bind DN that has write permission for the target DN.
`passwd="`*password*`"`	Specify the password for the bind DN. CAUTION: the password is visible in this tag, so the JSP file should be read-protected.
`dn="`*newDN*`"`	Specify the full DN of the target entry to be modified.
`mods="`*varName*`"`	Give the name of a variable containing the set of modifications to perform. When this attribute is omitted, the `<ldap:modify>` tag must contain the `<ldap:mod>` definitions.
`response="`*varName*`"`	Give the name of a variable (in the PAGE scope) for storing any error response. When omitted, the default name is `response`. If an LDAP error occurs during the operation, this variable will be set to the corresponding `LDAPException` Bean.

EXAMPLES

1. The following example shows individual modifications that are referenced by the `mods` attribute of the `<ldap:modify>` element:

```
<ldap:mod var="mods" scope="request"
           op="replace" name="postalAddress"
           value="123 Main Street"/>

<ldap:mod var="mods" scope="request"
           op="replace" name="facsimileTelephoneNumber"
           value="408 555 1234"/>
```

```
<ldap:mod var="mods" scope="request"
           op="delete" name="telephoneNumber"
           value="407 555 9876"/>

<ldap:mod var="mods" scope="request"
           op="add" name="telephoneNumber"
           value="408 555 6789"/>

<ldap:modify host="phonebook.siroe.com"
              user="cn=Directory Manager" passwd="password"
              dn="uid=6,ou=People,dc=siroe,dc=com">
              mods="$request:mods" response="modifyError/>
```

2. The second example performs the same operation, but the modifications are defined within the <ldap:add> tag. Note that the default modification is to replace an attribute value, so the op attribute may be omitted:

```
<ldap:modify host="phonebook.siroe.com"
              user="cn=Directory Manager" passwd="password"
              dn="uid=6,ou=People,dc=siroe,dc=com"
              response="modifyError">
    <ldap:mod name="postalAddress" value="123 Main Street"/>
    <ldap:mod name="facsimileTelephoneNumber" value="408 555 1234"/>
    <ldap:mod op="delete" name="telephoneNumber"
               value="407 555 9876"/>
    <ldap:mod op="add" name="telephoneNumber"
               value="408 555 6789"/>
</ldap:modify>
```

SEE ALSO

<ldap:mod>

<ldap:search>

Performs an LDAP search and returns the set of resulting entries. This element is an iterator that will evaluate its contents for every entry in the search results.

DESCRIPTION

The <ldap:search> element searches the directory according to the specified base, search scope, and filter. There are two ways to process the search results:

- Use other elements inside this tag to process each entry. The tag is an iterator and will evaluate its contents for each of the entries in the results.
- Specify a variable name and optional scope for storing the search results. Then access this variable from other elements to process the entries it contains.

In both cases, the results may be accessed as variables or using the <ldap:dn> and <ldap:attr> tags to extract information.

SYNTAX

```
<ldap:search [host="hostname" port="port" ]
             [user="bindDN" passwd="password" ]
             base="baseDN" [srchScope="base|one|sub" ]
             [filter="filterString" attrs="attrList" ]
             [maxEntries="number" controls="beanName" ]
             [entry="iterName" response="varName" ]>
    <...> ...
</ldap:search>
```

```
<ldap:search [host="hostname" port="port" ]
             [user="bindDN" passwd="password" ]
             base="baseDN" [srchScope="base|one|sub" ]
             [filter="filterString" attrs="attrList" ]
             [maxEntries="number" controls="beanName" ]
             var="varName" [scope="varScope" response="varName" ]/>
```

ATTRIBUTES

host="*hostname*"	Specify the hostname of the directory server. When this attribute is omitted, the default is localhost.
port="*port*"	Specify the port number of the directory server on the given host. When this attribute is omitted, the default port is 389.
user="*bindDN*"	Specify a bind DN for accessing your directory. If the bind DN and its password are omitted, the operation will use anonymous binding. The bind DN determines what entries and attributes will appear in the search results, according to the DN's access permissions.
passwd="*password*"	Specify the password for the bind DN. **CAUTION: the password is visible in this tag, so the JSP file should be read-protected.**
base="*baseDN*"	Specify the base DN for the search operation.
srchScope="*scope*"	Specify the scope of a search with one of the following values: • base - For searching only the base entry. This is the default. • one - For searching only the children of the base entry. • sub - For searching the base entry and all its descendants.
filter="*filterString*"	Specify an LDAP filter string for the search. When omitted, the default filter is (objectclass=*), which returns all entries in the search scope.
attrs="*attrList*"	Specify the list of attributes that will be returned with their values for each entry in the search results. Attribute names in the list are separated by vertical bars (|).

`maxEntries="`*number*`"`	Specify the maximum number of entries to return. A search that yields more than this limit will generate an exception. Regardless of the value specified here, if at all, a search will never return more entries than the number allowed by the directory server's `nsslapd-sizelimit` attribute whose default is 2,000 entries. See "nsslapd-sizelimit (Size Limit)" in the *Sun ONE Directory Server Reference Manual*.
`entry="`*iterName*`"`	Give the name of the loop variable used to store each entry in turn when iterating through the search results. When this attribute is omitted, the default name of the loop variable is `entry`.
`var="`*varName*`"`	Give the name of the variable where the search results will be stored. This variable name and scope may be omitted when search results are processed in the element contents.
`scope="`*varScope*`"`	Give the scope of the variable where the search results will be stored. When omitted, the default scope is PAGE.
`controls="`*beanName*`"`	Give the name of a variable containing the LDAP controls for this search. See `<ldap:sortControl>` and `<ldap:vlControl>`.
`response="`*varName*`"`	Give the name of a variable (in the PAGE scope) for storing any error response. When omitted, the default name is `response`. If an LDAP error occurs during the operation, this variable will be set to the corresponding `LDAPException` Bean.

EXAMPLES

1. This first example shows a simple search that displays the DN and several attributes for each result. The comments in the code explain the different ways to access attribute information in the results.

```
<ldap:search host="phonebook.example.com"
             user="cn=Directory Manager" passwd="password"
             base="ou=People,dc=example,dc=com" srchScope="sub"
             filter="(surname=a*)"
             attrs="dn|cn|employeeNumber|mail|telephonenumber">

   <!-- "entry" is the default iterator variable for results -->
   Entry: <jx:expr value="$entry"/><br>

   <!-- The DN can be read from the entry structure -->
   DN: <jx:expr value="$entry.dn"/><br>

   <!-- The pageContext Bean contains all attribute values -->
   cn: <%= ((EntryBean) pageContext.
             getAttribute("entry")).getAttr("cn").getValue() %><br>

   <!-- ldap:attr will display the value of the attribute -->
   ID: <ldap:attr entry="$entry" name="employeeNumber"/><br>
```

```
    <!-- ldap:attr can also store the value of the attribute, for
         use in other expressions -->
    <ldap:attr entry="$entry" name="mail" var="address" default=""/>
    mail: <jx:if test="$address != ''">
              <A HREF=mailto:<jx:expr value="$address"/>
              ><jx:expr value="$address"/></A>
          </jx:if><br>

    <!-- For multivalued attributes, use ldap:attr to retrieve all
         values, then iterate -->
    <ldap:attr entry="$entry" name="telephonenumber" op="values"
               var="values" scope="sess"/>
    <jx:forEach var="val" items="$session:values">
        telephonenumber: <jx:expr value="$val"/>
    </jx:forEach><p>

</ldap:search>
```

2. The next example performs an anonymous search and retrieves all attributes of the matching entries. It uses iterators inside the `<ldap:search>` element to display them all in the search results.

```
<ldap:search host="phonebook.example.com"
             base="ou=People,dc=example,dc=com" srchScope="sub"
             filter="(surname=b*)" var="results">

    Entry: <jx:expr value="$entry"/><br>
    DN: <jx:expr value="$entry.dn"/><br>
    Number of attributes:
        <jx:expr value="$entry.size" default="NULL"/><P>

    <table border="0" cellpadding="0" cellspacing="2">
        <jx:forEach var="oneAttr" items=$entry><tr>

            <td valign="top" align="right" nowrap>
                <jx:expr value="$oneAttr.name">
            </td>

            <td valign="top">
                <jx:forEach var="oneVal" items="$oneAttr">
                    <jx:expr value="$oneVal"><br>
                </jx:forEach>
            </td>

        </tr></jx:forEach>
    </table>
</ldap:search>

<P>Number of entries returned:
    <jx:expr value="$results.size" default="NULL"/><br><br>
```

3. The following example stores search results in a variable and accesses them outside of the <ldap:search> element. It also shows all information that is available for each attribute retrieved in a search. The search itself returns all configuration attributes under the configuration root entry.

```
<ldap:search host="phonebook.example.com"
              user="cn=Directory Manager" passwd="password"
              base="o=Root" srchScope="one"
              var="result"/>

<util:logScopes/>

<jx:forEach var="entry" items="$result">
    <!-- Search scope one should return only a single entry -->
    Entry: <jx:expr value="$entry"/><br>
    DN: <jx:expr value="$entry.dn""/><br>

    <jx:forEach var="oneAttr" items="$entry">
        Name=<jx:expr value="$oneAttr.name"/><br>
        <jx:forEach var="oneVal" items="$oneAttr">
            Value=<jx:expr value="$oneVal"/><br>
        </jx:forEach>
        Single.Value=<jx:expr value="$attr.value"
                              default="NULL"/><br>
        Size=<jx:expr value="$attr.size"/><br>
        Subtype=<jx:expr value="$attr.subtype" default="NULL"/><br>
        Values=<jx:expr value="$attr.values" default="NULL"/><br>
        ByteValue=<jx:expr value="$attr.byteValue"
                           default="NULL"/><br>
        ByteValues=<jx:expr value="$attr.byteValues"
                             default="NULL"/><br>

  </jx:forEach>

</ldap:search>
```

4. The last example produces an exception by setting a low limit to the number of entries returned and then displays all information about the exception.

```
<ldap:search host="phonebook.example.com"
              base="ou=People,dc=example,dc=com" srchScope="sub"
              maxEntries="3" response="searchError">
    DN=<jx:expr value="$entry.dn" default="NULL"/>
</ldap:search>

Response:
    <jx:expr value="$searchError" default="NULL"/><br>
Response.code:
    <jx:expr value="$searchError.code" default="NULL"/><br>
```

```
    Response.exception:
        <jx:expr value="$searchError.exception" default="NULL"/><br>
    Response.message:
        <jx:expr value="$searchError.message" default="NULL"/><br>
    Response.klass:
        <jx:expr value="$searchError.klass" default="NULL"/><br>
```

SEE ALSO

<ldap:attr>, <ldap:sortControl>,
<ldap:sortControl>, <ldap:vlControl>, <ldap:vlControl>

<ldap:sortControl>

Define an LDAPSortControl for sorting search results. This element must contain <ldap:sortKey> elements to specify the keys.

DESCRIPTION

Use the <ldap:sortControl> and <ldap:sortKey> elements together to specify one or more attributes to use for sorting search results. The first <ldap:sortKey> element it contains will be the primary sort key, the second one will be the secondary sort key, and so on.

Specify a variable name and optional scope to store the control definition and then reference this variable with the controls attribute of the <ldap:search> element. You may add the sort controls to an existing variable, as long as it is an instance of a ControlsBean, created by either <ldap:sortControl> or <ldap:vlControl>.

SYNTAX

```
<ldap:sortControl var="varName" [ scope="scopeName" ]>
    <ldap:sortKey ...> ...
</ldap:sortControl>
```

ATTRIBUTES

var="*varName*"	Give the name of the variable where the sort control will be stored.
scope="*varScope*"	Give the scope of the variable where the sort control will be stored.

EXAMPLES

```
<ldap:sortControl var="controls">
    <ldap:sortKey key="sn"/>
    <ldap:sortKey key="givenname"/>
</ldap:sortControl>
```

```
<ldap:search host="phonebook.example.com"
             base="ou=People,dc=example,dc=com" srchScope="sub"
             filter="(surname=a*)" controls="$controls"
             attrs="dn|sn|givenname|employeeNumber|telephonenumber"
             var="results" scope="page" response="searchError"/>
```

SEE ALSO

<ldap:vlControl>, <ldap:search>

<ldap:sortKey>

Defines the attribute to use for sorting within an <ldap:sortControl> element.

DESCRIPTION

The <ldap:sortKey> element may only occurs within an <ldap:sortControl> element. It specifies an attribute to use as the key for sorting the results of a search, and optionally reverse sort order or a matching rule.

SYNTAX

<ldap:sortKey key="*attrName*" [reverse="yes" matchRule="*OID*"]/>

ATTRIBUTES

key="*attrName*"	Specify the name of the attribute to use as the sort key.
reverse="yes"	Specify reverse sort order on the given key. When omitted, forward sort order is used by default.
matchRule="*OID*"	Specify the OID of a matching rule applicable to the attribute key.

SEE ALSO

<ldap:sortControl> and its examples.

<ldap:vlControl>

This element may be used in conjunction with <ldap:sortControl> to perform searches with sorted virtual lists (VL).

DESCRIPTION

The <ldap:vlControl> element allows you to add virtual list (VL) functionality to sorted searches. Use in conjunction with the <ldap:sortControl> by specifying the same name and optional scope for the var and scope attributes.

DESCRIPTION

The <ldap:vlControl> element allows you to add virtual list (VL) functionality to sorted searches. Use in conjunction with the <ldap:sortControl> by specifying the same name and optional scope for the var and scope attributes.

SYNTAX

For the initial search:

```
<ldap:vlControl jump="string" before="number" after="number"
                var="varName" [ scope="scopeName" ]/>
```

For subsequent searches in the same virtual list:

```
<ldap:vlControl start="index" before="number" after="number"
                content="number" var="varName" [ scope="scopeName" ]/>
```

ATTRIBUTES

jump="*string*"	Specify a string value or partial value for the first VL search. This string will be matched against the value of the sort key attribute to determine the initial index of the virtual list.
start="*index*"	Specify the index of the target entry in the virtual list during subsequent searches.
before	Specify the number of entries preceding the indexed entry to return in the VL search results.
after	Specify the number of entries following the indexed entry to return in the VL search results.
content	The number of entries in the complete search results. This number is returned during the first search and should be sent with subsequent searches of the same virtual list.
var="*varName*"	Give the name of the variable where the VL control will be stored.
scope="*varScope*"	Give the scope of the variable where the VL control will be stored.

EXAMPLES

```
<ldap:sortControl var="controls">
    <ldap:sortKey key="cn"/>
</ldap:sortControl>

<ldap:vlControl jump="A" before="5" after="5" var="controls"/>

<ldap:search host="phonebook.example.com"
             base="ou=People,dc=example,dc=com" srchScope="sub"
             filter="(surname=a*)" controls="$controls"
             var="results" scope="page" response="searchError"/>
```

```
VL Response = <jx:expr value="$results.vlResponse" default="NULL"/>

<table border="2">
    <jx:forEach var="entry" items="$results">
        <tr><td><ldap:attr entry="$entry" name="cn"/></td></tr>
    </jx:forEach>
</table>

<!-- Use the VL info from the first search to prepare subsequent
      searches of the same list. -->
<jx:set var="first" value="$results.vlResponse.firstPosition"/>
<jx:set var="count" value="$results.vlResponse.contentCount"/>

<% Integer firstI = (Integer) pageContext.getAttribute("first");
    Integer countI = (Integer) pageContext.getAttribute("count");
    if ((firstI != null) && (countI != null)) {
        int next = firstI.intValue() + 10;
        if (next <= countI.intValue())
            application.setAttribute("next", new Integer(next));
        else
            application.removeAttribute("next");
    }
    application.removeAttribute("last");
%>
```

SEE ALSO

`<ldap:sortControl>`, `<ldap:search>`

Utilities Tag Library

The Utilities Tag Library is also built on JSPTL and provides some useful elements for writing LDAP-enabled JSP applications. These tags perform such tasks as reading values from a properties file, testing the existence of an attribute, and so on. The tag library definition is contained in the `util.tld` found in the *DSRK_base*`/jdgw/phonebook-app/WEB-INF` directory.

<util:date>

Outputs a date and time string using the `java.text.DateFormat` class.

DESCRIPTION

The `<util:date>` element generates a date and time string according to the given locale. For a complete description of the possible date formats, see the API reference for the `java.text.DateFormat` class.

By default the element will display the current date in the page output. However, you may specify a different date string to reformat, and you may give the name of a variable and optional scope to store the formatted string.

SYNTAX

```
<util:date [ value="timeString" format="SHORT|MEDIUM|LONG|FULL" ]
           [ lang="subtype" cntry="subtype" varnt="localeVariant" ]
           [ var="varName" scope="varScope" ]/>>
```

ATTRIBUTES

`value`	Give a date and time string to be reformatted. This input must be an ISO 2014 timestamp (*YYYYMMDDHHmmss*Z) or in a format recognized by the `java.text.DateFormat.parse` method. When this attribute is omitted, the tag will output the current date.
`format`	Specify the format of the output date and time string. The possible values are the same as the date format constant fields defined in `java.text.DateFormat`. The format is locale dependent, but generally, the formats have the following characteristics: • `SHORT` - Completely numeric, for example 02/02/02 or 12:34pm. • `MEDIUM` - Uses abbreviations: Feb 02, 2002; this is the default. • `LONG` - Spells out the date and includes times with seconds. • `FULL` - Includes day of the week and time zone.

`lang="`*subtype*`"`
`cntry="`*subtype*`"`
`varnt="`*localeVariant*`"`
	Specify the language and country subtypes that define a locale. You must specify a language and country together, and you may also give an optional locale variant. These three attributes determine the locale for the date format. When omitted, the default locale is used.

`var="`*varName*`"`	Give the name of the variable where the date string will be stored.
`scope="`*varScope*`"`	Give the scope of the variable where the date string will be stored. When omitted, the default scope is `PAGE`.

EXAMPLES

```
SHORT: <util:date format="short"/>
MEDIUM: <util:date format="medium"/>
LONG: <util:date format="long"/>
FULL: <util:date format="full"/><P>
DEFAULT: <util:date/><P>
TIMESTAMP 20010807163841Z:
    <util:date format="full" value="20010807163841Z"/>
```

<util:encode>

Encode a string into UTF-8 format.

DESCRIPTION

This element converts its contents or its `value` attribute to UTF-8, where all characters are encoded in 8 bits.

If you do not specify a variable and optional scope in which to store the encoded value, it will be displayed as part of the page output.

SYNTAX

`<util:encode [ var="`*varName*`" scope="`*varScope*`" ]>`*inputString*`</util:encode>`

`<util:encode value="`*inputString*`" [ var="`*varName*`" scope="`*varScope*`" ]/>`

ATTRIBUTES

`value="`*inputString*`"`	Specify the value to be encoded.
`var="`*varName*`"`	Give the name of the variable where the encoded value will be stored.
`scope="`*varScope*`"`	Give the scope of the variable where the encoded value will be stored. When omitted, the default scope is PAGE.

EXAMPLES

```
DN string: "cn=Babs Jensen,ou=People,dc=example,dc=com"
Normalized:
    <ldap:dn value="cn=Babs Jensen,ou=People,dc=example,dc=com"/>
<ldap:dn value="cn=Babs Jensen,ou=People,dc=example,dc=com" var="dn"/>
Encoded: <util:encode value="$dn"/>
```

<util:importParams>

Import all parameters from the ServletRequest into the REQUEST scope.

DESCRIPTION

This element reads all the parameters of the request and sets page context attributes, either single or multivalued, in the REQUEST scope.

SYNTAX

```
<util:importParams/>
```

<util:log>

Writes a string to the log file.

DESCRIPTION

This element will add the following line to the ServletContext log:

<Log> *logString*

SYNTAX

```
<util:log value="logString"/>
```

ATTRIBUTES

value="*logString*"	Give the string value to be written in the log.

EXAMPLES

```
<util:log value="Log this value"/>
```

<util:logScopes>

Logs all variables and values defined currently defined in all scopes.

SYNTAX

```
<util:logScopes/>
```

DESCRIPTION

This element will write a section in the `ServletContext` log containing the name and value of all variables or attributes in a specified scope.

SEE ALSO

`JSP Standard Tag Library`

<util:present>

Checks the existence of a variable and returns `true` if present.

DESCRIPTION

This element checks for the existence of a variable name in a given scope and sets a boolean value in another variable. The result is `true` if the named variable exists and `false` otherwise. The scope attribute of this tag determines the scope of the variable name to test, not the scope of the output variable.

SYNTAX

`<util:present name="`*varName*`" [ scope="`*nameScope*`" var="`*varName*`" ]/>`

ATTRIBUTES

`name="`*varName*`"`	Specify the name of the variable of which to test the existence.
`scope="`*nameScope*`"`	Specify the scope in which to find the variable to test. When omitted, the variable name is assumed to be in the PAGE scope.
`var="`*varName*`"`	Specify the name of the variable where the boolean result will be stored. When omitted, the variable will be called `present`.

<util:prop>

Retrieve a property value from a properties file.

DESCRIPTION

This element will retrieve a value corresponding to the given property key from a properties file. You may specify the language, country, and variant that determine a locale through the `lang`, `cntry`, and optional `varnt` attributes.

The name of the properties bundle and optional locale determine the name of the file where the keys and values are defined. The element will return an error if the file cannot be found.

The property key is the name of the property, as found in the file. You may also specify an optional default value that will be returned when the key does not exist in the file.

If you do not specify a variable and optional scope in which to store the property value, it will be displayed as part of the page output.

SYNTAX

```
<util:prop prps="bundleName" key="propName" [ def="value" ]
           [ lang="subtype" cntry="subtype" varnt="localeVariant" ]
           [ var="varName" scope="varScope" ]/>
```

ATTRIBUTES

`prps="`*`bundleName`*`"`	Specify the name of the properties bundle, which is the same as the properties file name without the `.properties` suffix.
`key="`*`propName`*`"`	Specify the name of the key name of the value to be retrieved.
`def="`*`value`*`"`	Give a default value that will be returned if the key is not defined in the properties bundle.

`lang="`*`subtype`*`"`
`cntry="`*`subtype`*`"`
`varnt="`*`localeVariant`*`"`
	Specify the language and country subtypes that define a locale. You must specify a language and country together, and you may also give an optional locale variant. These three attributes determine which properties file to access. When omitted, the properties for the default locale are used.

`var="`*`varName`*`"`	Give the name of the variable where the property value will be stored.
`scope="`*`varScope`*`"`	Give the scope of the variable where the property value will be stored. When omitted, the default scope is PAGE.

EXAMPLES

The following example from the LookMeUp application uses localized properties to create a row of headings in a table output.

```
<table border="0" cellpadding="1" cellspacing="0" width="100%"
       bgcolor="#FFFFFF" align="center">
   <tr align="left" bgcolor="#990000"> <!-- Shaded heading row -->

       <td><b>
           <util:prop prps="screen" key="searchresults.name"
                      lang="$app:language" cntry="$app:country"/>
       </b></td>
```

```
        <td><b>
            <util:prop prps="screen" key="searchresults.phone"
                       lang="$app:language" cntry="$app:country"/>
        </b></td>
        <td><b>
            <util:prop prps="screen" key="searchresults.email"
                       lang="$app:language" cntry="$app:country"/>
        </b></td>
    </tr> …
</table>
```

JSP Standard Tag Library

JSPTL, Early Access Release 1.2, is the standard JavaServer Pages Tag Library. It contains simple tags that provide core functionality common to many JSP applications.

NOTE While the JSPTL has been superseded by the JSP Standard Tag Library (JSTL), the LookMeUp application relies on the older JSPTL described in this section. If you develop new JSP, we recommend you use the latest JSTL. The latest release and its documentation is available from:

`http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html`

JSPTL is comprised of two tag libraries. The two libraries contain very common sets of tags, with one major difference: the first library (`jr`) has tags that accept request-time attribute values (`rtexprvalues`) for their attributes, while the second library (`jx`) accepts attribute values specified using the *expression language* (EL) introduced in JSPTL. To use the `jr` tags, include the following line at the top of your `.jsp` file:

```
<%@ taglib uti=http://java.sun/com/jsptl/ea/jr prefix="jr" %>
```

NOTE You may choose any prefix, but `jr` is the current recommendation for the `rtexprvalue` version of the JSPTL library.

The LookMeUp application included with the DSRK relies on the `jx` library that supports EL. This second library is detailed in this guide. To access this library, the application's JSP use the following directive:

```
<%@ taglib uti=http://java.sun/com/jsptl/ea/jx  prefix="jx" %>
```

The tag library definition is contained in the `jx.tld` found in the *DSRK_base*`/jdgw/phonebook-app/WEB-INF` directory.

<jx:expr>

Returns the value of its expression.

SYNTAX

`<jx:expr value="`*valueExpr*`" [ default="`*value*`" ]/>`

ATTRIBUTES

`value="`*valueExpr*`"`	Specify an expression that will be the value to return. Use the $*varName* syntax to retrieve the value of a variable.
`default="`*value*`"`	Give a default value to be returned if the variable does not exist or has a null value.

DESCRIPTION

Use this tag to fetch the value of a variable by specifying the $*varName* syntax in the value attribute. This element is the EL-equivalent of the <%=...%> element.

EXAMPLE

See `<jx:set>` and its example.

<jx:set>

Store the result of an expression evaluation in a scoped variable.

SYNTAX

`<jx:set var="`*varName*`" [ scope="`*varScope*`" ] value="`*valueExpr*`"/>`

`<jx:set var="`*varName*`" [ scope="`*varScope*`" ]>`*contents*`</jx:set>`

ATTRIBUTES

`value="`*valueExpr*`"`	Specify an expression that will be the value to store. Use the $*varName* syntax to specify the value of a variable.
`var="`*varName*`"`	Give the name of the variable where the value will be stored.

scope="*varScope*"	Give the scope of the variable where the value will be stored. When omitted, the default scope is PAGE.

DESCRIPTION

Use this tag to evaluate an expression in the value attribute and store its result in the designated variable and optional scope. You may also store the elements which are contents of this tag in the designated variable.

EXAMPLE

```
<jx:set var="city" value="$custmer.address.city"/>
<jx:set var="customerFmt" scope="request">
    <font color="red">
        <jx:expr value="$city"/>
    </font>
</jx:set>
```

<jx:declare>

Declares a scripting variable.

SYNTAX

<jx:declare id="*varName*" [type="*typeClass*"]/>

ATTRIBUTES

id="*varName*"	Specify the name of the scripting variable.
type="*typeClass*"	Specify the type or Java class of the attribute. When omitted, the default type is java.lang.Object.

DESCRIPTION

Declares a scripting variable, initially defined by an existing scoped attribute of the same name.

EXAMPLE

```
<jx:declare id = "customer" type= "acme.Cutomer" />
```

<jx:forEach>

The basic iteration tag, accepting many different collection types.

SYNTAX

```
<jx:forEach var="iterName" items="varName" [ status="varName" ]
            [ begin="index" end="index" step="number" ]>
    contents
</jx:forEach>
```

ATTRIBUTES

`var="`*iterName*`"`	Specify the name of the variable in which to store the current iteration value.
`items="`*varName*`"`	Give the name of a variable containing values on which to iterate. With each iteration, the variable named by the `var` attribute will take the value of one of these items.
`status="`*varName*`"`	Specify the name of a variable from which loop status information, such as the current index, may be read.
`begin="`*index*`"`	Specify the index of the first element to process. The *index* value must be greater than or equal to 0. When omitted, iteration will start with the first element of the `items` variable.
`end="`*index*`"`	Specify the index of the last element to process. The *index* value must be greater than or equal to the `begin` index. When omitted, iteration will end with the last element of the `items` variable.
`step="`*number*`"`	Specify the number steps to advance in the list of `items` with each iteration. The step *number* must be greater than or equal to 1. The default step is 1. For example, `step="2"` will process every other element in the `items` variable.

DESCRIPTION

This element will process its contents once for every iteration defined by the `items` attribute and by the optional `begin`, `end`, and `step` attributes. During the iteration, the variable named by the `var` attribute will contain each value of the items in sequence. For example, this element allows you to process each value of a multi-valued LDAP attribute.

EXAMPLE

```
<jx:forEach var="customer" items="$customers">
    <jx:expr value="$customer.name">
</jx:forEach>
```

<jx:forToken>

Iterates over tokens in a string, separated by the specified delimiters.

SYNTAX

```
<jx:forToken var="iterName" items="tokenString" delims="delimChars"
             [status="varName" begin="index" end="index" step="number"]>
    contents
</jx:forToken>
```

ATTRIBUTES

`var="`*iterName*`"`	Specify the name of the variable in which to store the current iteration value.
`items="`*tokenString*`"`	Give a string containing the tokens on which to iterate. With each iteration, the variable named by the `var` attribute will take the value of each token.
`delims="`*delimChars*`"`	The set of characters that delimit the tokens in the `tokenString`. Each character in this string is a separator.
`status="`*varName*`"`	Specify the name of a variable from which loop status information, such as the current index, may be read.
`begin="`*index*`"`	Specify the index of the first token to process. The *index* value must be greater than or equal to 0. When omitted, iteration will start with the first token of the `items` variable.
`end="`*index*`"`	Specify the index of the last token to process. The *index* value must be greater than or equal to the `begin` index. When omitted, iteration will end with the last token of the `items` variable.
`step="`*number*`"`	Specify the number steps to advance in the list of `items` with each iteration. The step *number* must be greater than or equal to 1. The default step is 1. For example, `step="2"` will process every other token in the `items` variable.

DESCRIPTION

This element will process its contents once for every token defined by the `items` and `delims` attributes. As with the `<jx:forEach>` element, you may modify the iteration sequence with the optional `begin`, `end`, and `step` attributes. During the iteration, the variable named by the `var` attribute will contain each token in sequence. For example, this element allows you to process strings containing fields or choice values.

EXAMPLE

```
<jx:forToken var="token" items="blue|white|red" delims="|">
    contents
</jx:forToken>
```

<jx:if>

The basic conditional tag.

SYNTAX

```
<jx:if test="boolExpr" [ var="varName" scope="varScope" ]>
    contents
</jx:if>
```

ATTRIBUTES

`test="`*boolExpr*`"`	Specify a boolean expression that will evaluate to `true` or `false`.
`var="`*varName*`"`	Give the name of the variable where the boolean test result will be stored.
`scope="`*varScope*`"`	Give the scope of the variable where the boolean will be stored. When omitted, the default scope is PAGE.

DESCRIPTION

The `<jx:if>` element evaluates its *contents* if the boolean expression in the `test` attribute evaluates to `true`. The optional `var` attribute names a variable to store the Boolean result of evaluating the test condition.

EXAMPLES

```
<jx:if test="$customer.address.country == 'USA'"
        var="isUsCustomer">
    contents
</jx:if>
```

SEE ALSO

`<jx:choose>` `<jx:when>` `<jx:otherwise>`

<jx:choose> <jx:when> <jx:otherwise>

Conditional tag that allows mutually exclusive conditional operations marked by `<jx:when>` and `<jx:otherwise>`.

SYNTAX

```
<jx:choose>
    <jx:when test="boolExpr">
        contents
    </jx:when>
    ...
```

```
    [ <jx:otherwise>
        contents
    </jx:otherwise> ]
</jx:choose>
```

ATTRIBUTES

`test="`*boolExpr*`"`	Specify a boolean expression that will evaluate to `true` or `false`.

DESCRIPTION

The `<jx:choose>`, `<jx:when>`, and `<jx:otherwise>` elements allow the common `if-then-elsif-else` conditional construct.

The `<jx:choose>` element must directly contain one or more `<jx:when>` elements. It may directly contain an optional `<jx:otherwise>` element which must occur last. The boolean expressions in the `test` attributes will be evaluated in the order they appear and the *contents* of the first one that is `true` will be evaluated. If all are `false`, the contents of the `<jx:otherwise>` element will be evaluated, if present.

EXAMPLE

```
<ldap:search host="phonebook.example.com"
             base="ou=People,dc=example,dc=com" srchScope="sub"
             var="results" response="searchError"/>

<jx:choose>
    <jx:when test="$searchError == 'NULL'">
        <!-- process the search results -->
        <jx:choose>
            <jx:when test="$results.size > 1">
                <!-- display a table of search results-->
                <%@ include file="searchresults.jsp" %>
            </jx:when>

            <jx:when test="$results.size == 1">
                <!-- extract the single result from the list -->
                <jx:forEach var="result" items="$results">
                    <!-- set the request:dn for use in person.jsp -->
                    <jx:set var="dn" scope="request"
                            value="$result.dn"/>
                    <jsp:include page="person.jsp"></jsp:include>
                </jx:forEach>
            </jx:when>
            <jx:otherwise>
                <!-- Display a message -->
```

```
                <h1><center>No matching entry was found</h1>
            </jx:otherwise>
        </jx:choose>
    </jx:when>

    <jx:otherwise>
        <!-- display the search error -->
        Response: <jx:expr value="$resp" default="NULL"/><br>
        Message:
            <jx:expr value="$resp.message" default="NULL"/><br>
    </jx:otherwise>
</jx:choose>
```

Part 8

Additional Tools and Information

Chapter 34

NameFinder Application

NameFinder is a web-based tool to look up people in an Lightweight Directory Access Protocol (LDAP) database. This chapter provides information on NameFinder. It contains the following sections:

- Overview
- Deploying NameFinder
- Configuring NameFinder
- Search Parameters

Overview

NameFinder shows all relevant LDAP entries in an LDAP database based on certain search parameters. In order to run the application, you need either Sun™ ONE Web Server 6.0 SP5 or higher or Sun ONE Application Server 7.0 or higher to deploy it. The Java Development Kit (JDK) 1.4.0 or higher must be installed on the server. The WAR file and related files can be found in *DSRK_base*/NameFinder.

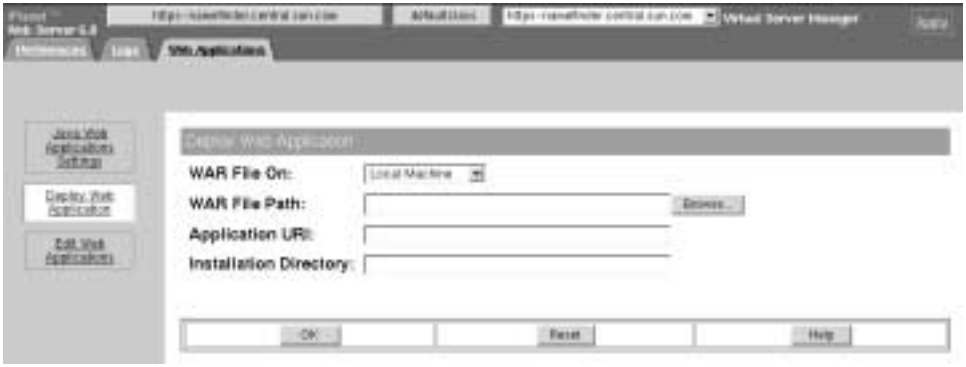
CAUTION The Java Development Kit (JDK) 1.4.0 or higher is mandatory. The default JDK for Sun ONE Web Server 6.0 is JDK 1.3.0 and NameFinder will not run on this.

Deploying NameFinder

Before you can deploy a web application manually, you must make sure that the *web_server_base*/bin/https/httpsadmin/bin directory is in your path and the IWS_SERVER_HOME environment variable is set to your *web_server_base* directory.

Sun ONE Web Server 6.0 SP5 has a web interface that can be used to easily deploy a web archive (WAR) file. Figure 34-1 on page 364 is a screen shot of the WAR deployment interface.

Figure 34-1 Web Server Interface For WAR Deployment



More complete information on how to deploy a WAR file using the interface or via the command line can be found in the documentation for Sun ONE Web Server 6.0.

Configuring NameFinder

After deploying the application, you need to modify the `NameFinder.properties` file. The NameFinder servlet (as defined in the file itself) retrieves this file automatically for configuration information to display entries from your directory. `NameFinder.properties` is located in *DSRK_base*`/NameFinder/WEB-INF/classes`. The following sections detail the information defined in `NameFinder.properties`. The information and attribute / value pairs are taken from the `sample.properties` file also included in *DSRK_base*`/NameFinder/WEB-INF/classes`. `sample.properties` was created for documentation purposes.

NOTE Everything in this file is case-sensitive.

Server Configuration Attributes

Information concerning the directory server and directory tree is defined in `NameFinder.properties`. This includes the base DN, the server domain and the version of LDAP used by the server. The server configurations are:

```
NameFinder.ldapBase=ou=people,dc=siroe,dc=com
NameFinder.ldapServers=directory.siroe.com
NameFinder.ldapVersion=2
```

(If no LDAP version is specified, the default is 3.)

```
NameFinder.servletName=NameFinder
```

(If no servlet name is specified, the default is `NameFinder`.)

```
NameFinder.ldapPort=389
```

(If no port is specified, the default is 389.)

```
NameFinder.ldapUser=
NameFinder.ldapPasswd=
```

NOTE Only simple authentication (anonymous or user-based) is supported. If you require SSL authentication, the connection must be provided by the servlet utilities.

LDAP Attribute Configurations

NameFinder can only obtain the LDAP attributes specified in this section of `NameFinder.properties`. The attribute fields are defined in ascending order. The syntax for the attribute configuration line is:

```
NameFinder.attr#=arg1|arg2|arg3|arg4|arg5
```

Where:

- The hash sign (#) represents the number given to the attribute / value pair. The numbers must be defined in ascending order. If an attribute number is missing, any attributes after the last valid sequential attribute are ignored.
- `arg1` denotes the search field option. Options are used in the search field to define a search for the particular attribute. For example, if you specify `f` for first name, you can search for a first name using `-f donald`. (Searching for donald without an option will generate results including first and last names; using the option restricts the results to first names only.) See “Search Parameters” on page 372 for more information.

CAUTION Do not use `F` as an option character. `F` is reserved for directly entering an LDAP search filter.

- arg2 defines the tablefield property. If more than one entry is found in a search, a table is generated for the results; each column is defined in the tablefield pairs listed in "Predefined Attribute Fields" on page 367. (The default table is specified in NameFinder.tablefields.) The table is sorted by the first entry. Clicking on the column header will refresh the table and sort it using the data in the column that is clicked.

NOTE A parameter is a name/value pair appended to the end of a URL. The parameter starts with a question mark (?) and takes the form name=value. NameFinder uses a &fields parameter to specify the order and columns displayed in a generated search results table. The arg2 values defined in this section are used with the &fields parameter. For example &fields=nfeP displays last name, first name, email and telephone number fields only.

- arg3 is the corresponding LDAP attribute.
- arg4 is the label displayed for the LDAP attribute if a single entry is shown.
- arg5 is the column label for the LDAP attribute in a table displayed if multiple entries are found. If this argument is not specified, the label from arg4 is used.

Name and Contact Information

```
NameFinder.attr1=n|n|sn|Name|Lastname

NameFinder.attr2=f|f|givenname||Firstname

NameFinder.attr3=|N|nickname|Nickname

NameFinder.attr4=|1|userdefinedattribute1|Title/Function

NameFinder.attr5=p|p|extensionphone|5digit Phone #|Phone

NameFinder.attr6=P|P|telephonenumber|Phone #

NameFinder.attr7=|A|altcontact|Admin Phone #

NameFinder.attr8=|F|facsimiletelephonenumber|FAX #

NameFinder.attr9=m|e|mail|eMail

NameFinder.attr10=|m|mobile|Mobile

NameFinder.attr11=|a|pager|Pager

NameFinder.attr12=|E|pageremail|SMS/Pager eMail|PagerEmail

NameFinder.attr13=||mailalternateaddress
```

Location Information

```
NameFinder.attr14=C|C|countrycode|Countrycode
NameFinder.attr15=l|l|globallocation|Building Code|LOC
NameFinder.attr16=|||Flexible Office|
NameFinder.attr17=g|g|l|Ext.Location
NameFinder.attr18=|M|extendedaddress|Mailstop
NameFinder.attr19=|r|roomnumber|Room #
```

Calendar Information

```
NameFinder.attr20=|c|calendar|Calendar
NameFinder.attr21=|Z|preferredtimezone|Local Time|Timezone
```

Miscellaneous Information

```
NameFinder.attr22=|h|labeleduri|Homepage
NameFinder.attr23=s|s|employeenumber|SunID
NameFinder.attr24=D|D|departmentnumber|Department #|CC
NameFinder.attr25=|||Direct Reports
NameFinder.attr26=r|R|reportsto|Reports to
NameFinder.attr27=|2|userdefinedattribute2|Personal Field2
NameFinder.attr28=u|u|uid|Loginname
NameFinder.attr29=||mailhost|Mailserver
NameFinder.attr30=||host|Hostname
NameFinder.attr31=||employeetype
NameFinder.attr32=||photourl
```

Predefined Attribute Fields

Predefined fields contain LDAP attributes as values. For example, if `lastNameField`, `firstNameField` and / or `emailField` are specified, NameFinder automatically searches on the attributes defined in these fields. It will search up to six levels to find an entry. Following is a list of the predefined fields.

- `uniqueIdField` determines which attribute is the unique ID.

```
NameFinder.uniqueIdField=employeenumber
```

CAUTION `uniqueIdField` must be specified. If not, NameFinder will not work. This LDAP field denotes a query which returns exactly ONE entry.

- If multiple entries are found during a search, the entries are generated in a table format. `tableFields` defines the column layout for the table.

  ```
  NameFinder.tableFields=sn|givenname|mail|employeenumber|extensionphone|
  globallocation|departmentnumber
  ```

  ```
  NameFinder.emailFields=mail|mailalternateaddress
  ```

- `reportsTableFields` defines the column layout of the Direct Reports table.

  ```
  NameFinder.reportsTableFields=sn|givenname|extensionphone|mail|globallo
  cation
  ```

- `teamTableFields` defines output fields for the Team Viewer. `firstNameField`, `lastNameField` and `locationField` are automatically gathered in addition to other attributes defined here. After setting up NameFinder, see the Online Help for instructions on how to set up and view a Team.

  ```
  NameFinder.teamTableFields=sn|telephonenumber|mail|calendar|labeleduri|
  mobile|pageremail
  ```

- `phoneFields` specifies all phone attributes. Searching on this field will invoke an `OR` search for all fields.

  ```
  NameFinder.phoneFields=extensionphone|telephonenumber|mobile
  ```

- `emailFields` specifies all email attributes. Searching on this field will invoke an `OR` search for all fields.

- `lastNameField` determines which attribute is the last name.

  ```
  NameFinder.lastNameField=sn
  ```

- `firstNameField` determines which attribute is the first name.

  ```
  NameFinder.firstNameField=givenname
  ```

- `emailField` determines which attribute is the email address.

  ```
  NameFinder.emailField=mail
  ```

CAUTION If you do not specify `lastNameField`, `firstNameField` and `emailField`, the default search, with no options specified, is on the `uniqueIdField`. If you require an additional search level, you can specify it as a predefined field with an LDAP filter.

`NameFinder.extraSearch=(|(cn=`*query_string*`*)(description=*`*query_string*`*)(global location=*`*query_string*`*)(streetaddress1=*`*query_string*`*)(streetaddress2=*`*query_string*`*)(city=*`*query_string*`*)(c=*`*query_string*`*))`

- `nickNameField` determines which attribute is the nickname.

  `NameFinder.nickNameField=nickname`

- `uidField` determines which attribute is the user ID.

  `NameFinder.uidField=uid`

- `photoUrlField` determines which attribute contains the URL to a photograph.

  `NameFinder.photoUrlField=photourl`

- `locationField` determines the location attribute.

  `NameFinder.locationField=globallocation`

- `calendarServerField` determines the calendar server.

  `NameFinder.calendarServerField=calendar`

- `calidField` determines the calendar ID.

  `NameFinder.calidField=uid`

- `titleField` determines which attribute contains the person's title.

  `NameFinder.titleField=userdefinedattribute1`

- `timeZoneField` determines which attribute is the time zone.

  `NameFinder.timeZoneField=preferredtimezone`

- `phoneField` determines which attribute contains the phone number.

  `NameFinder.phoneField=extensionphone`

- `mobilePhoneField` determines which attribute conains the mobile/cell phone number.

  `NameFinder.mobilePhoneField=mobile`

- `managerField` determines a person's direct reports.

  `NameFinder.managerField=reportsto`

CAUTION If you do not use the same value defined in `uniqueIdField` as the value defined in the `managerField`, you must set an additional predefined field: `managerSearchField`. For example, if your `uniqueIdField` is set to `uid`, and the `managerField` is set to `employeenumber`, you must set the `managerSearchField` for a correct search.

```
NameFinder.managerSearchField=employeenumber
```

- The following fields change the OrgTree view. Images will take much longer to build up the page.

```
NameFinder.emptyEntry=<img align="TOP" src="nf/empty.gif">

NameFinder.scanEntry=<img align="TOP" src="nf/scan.gif">

NameFinder.lastEntry=<img align="TOP" src="nf/lastEntry.gif">

NameFinder.entry=<img align="TOP" src="nf/entry.gif">
```

Code Example 34-1 is the `NameFinder.properties` file.

Code Example 34-1 NameFinder.properties File

```
NameFinder.ldapBase=ou=people,dc=siroe,dc=com
NameFinder.ldapServers=directory.siroe.com
NameFinder.ldapVersion=2
NameFinder.servletName=NameFinder
#NameFinder.ldapPort=389
#NameFinder.ldapUser=
#NameFinder.ldapPasswd=
#
#Name and contact information
#
NameFinder.attr1=n|n|sn|Name|Lastname
NameFinder.attr2=f|f|givenname||Firstname
NameFinder.attr3=|N|nickname|Nickname
NameFinder.attr4=|t|userdefinedattribute1|Title/Function
NameFinder.attr5=p|p|extensionphone|5digit Phone #|Phone
NameFinder.attr6=P|P|telephonenumber|Phone #
NameFinder.attr7=|A|altcontact|Admin Phone #
NameFinder.attr8=|F|facsimiletelephonenumber|FAX #
NameFinder.attr9=m|e|mail|eMail
NameFinder.attr10=|m|mobile|Mobile
NameFinder.attr11=|a|pager|Pager
NameFinder.attr12=|E|pageremail|SMS/Pager eMail|PagerEmail
#
#Location information
#
NameFinder.attr13=C|C|countrycode|Countrycode
NameFinder.attr14=l|l|globallocation|Building Code|LOC
NameFinder.attr15=|||Flexible Office|
NameFinder.attr16=g|g|l|Ext.Location
```

Code Example 34-1 NameFinder.properties File *(Continued)*

```
NameFinder.attr17=|M|extendedaddress|Mailstop
NameFinder.attr18=|r|roomnumber|Room #
#
#Calendar information
#
NameFinder.attr19=|c|calendar|Calendar
NameFinder.attr20=|Z|preferredtimezone|Local Time|Timezone
#
#Misc. information
#
NameFinder.attr21=|h|labeleduri|Homepage
NameFinder.attr22=s|s|employeenumber|SunID
NameFinder.attr23=D|D|departmentnumber|Department #|CC
NameFinder.attr24=||dummyfield1|Direct Reports
NameFinder.attr25=r|R|reportsto|Reports to
NameFinder.attr26=||userdefinedattribute2|Personal Field2
NameFinder.attr27=u|u|uid|Loginname
NameFinder.attr28=||mailhost|Mailserver
NameFinder.attr29=||host|Hostname
NameFinder.attr30=||employeetype
NameFinder.attr31=||photourl|Photo
#
NameFinder.tableFields=sn|givenname|mail|employeenumber|extensionphone|glob
allocation|departmentnumber
NameFinder.phoneFields=extensionphone|telephonenumber|mobile
NameFinder.reportsTableFields=sn|givenname|extensionphone|mail|globallocati
on
NameFinder.teamTableFields=sn|telephonenumber|mail|calendar|labeleduri|mobi
le|pageremail
NameFinder.lastNameField=sn
NameFinder.firstNameField=givenname
NameFinder.nickNameField=nickname
NameFinder.emailField=mail
NameFinder.uidField=uid
NameFinder.photoUrlField=photourl
NameFinder.locationField=globallocation
NameFinder.calendarServerField=calendar
NameFinder.calidField=uid
NameFinder.timeZoneField=preferredtimezone
NameFinder.uniqueIdField=employeenumber
NameFinder.phoneField=extensionphone
NameFinder.mobilePhoneField=mobile
NameFinder.managerField=reportsto
#managerSearchField needs to be set if your do not search with the
uniqueIdField in the managerField
NameFinder.managerSearchField=employeenumber
#NameFinder.extraSearch=(|(cn=@*)(description=*@*)(globallocation=*@*)(stre
etaddress1=*@*)(streetaddress2=*@*)(city=*@*)(c=*@*))
#
#NameFinder.emptyEntry=<img align="TOP" src="/nf/empty.gif">
#NameFinder.scanEntry=<img align="TOP" src="/nf/scan.gif">
#NameFinder.lastEntry=<img align="TOP" src="/nf/lastEntry.gif">
```

Code Example 34-1 NameFinder.properties File *(Continued)*

```
#NameFinder.entry=<img align="TOP" src="/nf/entry.gif">
#
```

Search Parameters

Table 34-1 is a listing of available search parameters based on `NameFinder.properties`. The parameters refer to the defined options as detailed in "LDAP Attribute Configurations" on page 365. Wildcards (*) are allowed in the search field.

Table 34-1 Search Parameters

Parameter	Example
firstname lastname	John Smith
lastname firstname	m* da silva
firstname lastname	de* be*
lastname	-n Smith
lastname	-n m*ller
firstname or lastname	m*cdonald
firstname or lastname	*ben
firstname	-f hugo
firstname lastname	d* sch*
SunID	12345
firstname.lastname	d*.sch*
-p phone	-p x62155*
-u loginname	-u scott
-l locationcode	-l muc07
URL	http://namefinder/nf/nfJon.txt

Chapter 35

Java Naming and Directory Interface

Java™ Naming and Directory Interface is an API used to provide naming and directory functionality to applications written in the Java programming language. This chapter provides information on how to the tool. It contains the following sections:

- Overview
- JNDI Service Provider for DSML
- JNDI Booster Pack for LDAP Service Provider
- Additional Information

Overview

Java Naming and Directory Interface (JNDI) is designed specifically for the Java platform using Java's object model. Using JNDI, Java applications can store and retrieve Java objects of any type. In addition, JNDI provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes.

JNDI is defined independent of any specific naming or directory service implementation, enabling applications to access different (or possibly multiple) naming and directory services with one API. Assorted naming and directory service providers can be plugged in behind this common API to access information from a variety of existing naming and directory services, including Lightweight Directory Access Protocol (LDAP), Novell® Directory Service (NDS), Domain Name System (DNS), and Network Information System / Yellow Pages (NIS / YP). In addition, JNDI enables the applications to coexist with legacy software and systems.

NOTE A *service provider* is basically a set of classes that implement various JNDI interfaces for a specific naming and directory service - much like a JDBC driver implements various JDBC interfaces for a particular database system.

JNDI Service Provider for DSML

The DSRK includes the Early Access 1 release of the JNDI DSML v2 Service Providers.

NOTE Directory Services Markup Language (DSML) is a markup language that enables you to represent directory entries and commands in XML.

The following components (and additional information including READMEs) can be found in *DSRK_base*/jndi-dsml.

- dsmlv2.jar - An archive of class files for the service providers and utilities.
- providerutil.jar - An archive of utilities used by service providers developed by Sun Microsystems. The DSML v2 service provider uses some of the classes in this archive.

CAUTION This archive file is NOT interchangeable with the providerutil.jar file that you might have downloaded with any other service providers from Sun Microsystems althouh it is safe for other providers to use this providerutil.jar.

- jndi-dsmlv2-ext.html - Installation instructions.
- jndi-dsmlv2.html - Documentation of the service providers and utilities.
- The Javadocs for the DSML v2 utility classes.
- Examples of how to use the service providers and utilities.

JNDI Booster Pack for LDAP Service Provider

The DSRK includes the 1.0 release of the JNDI LDAP Booster Pack. This software is NOT an LDAP service provider; it works in conjunction with an LDAP service provider. It must be installed into a Java 2 Standard Edition (J2SE) SDK or Runtime Environment (JRE) that already has a JNDI/LDAP service provider.

NOTE The recommended version of the J2SE platform to use with this release is 1.4.x.

The following components (and additional information including READMEs) can be found in *DSRK_base*/`jndi-ldap`.

- `ldapbp-install.html` - Installation instructions for the booster pack.
- `ldapbp-overview.html` - Documentation for the software in the booster pack.
- doc/ldapbp/api/ - Javadocs for the public classes in the booster pack.
- `ldapbp.jar` - An archive ("booster pack") for supporting various LDAP controls and extensions.

Additional Information

The following locations contain more information on JNDI.

The JNDI Tutorial

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

JNDI page for Java Developers

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

XML page for Java Developers

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

Javadocs for JNDI 1.2.1

http://java.sun.com/products/jndi/1.2/javadoc

Chapter 36

Attribute Value Uniqueness Plug-In

The attribute value uniqueness plug-in is an experimental feature that requires Sun™ ONE Directory Server Resource Kit (DSRK) 5.2. This server plug-in enforces the uniqueness of attribute values in replicas that participate in a multi-master replication scenario. This chapter provides information on the plug-in. It contains the following sections:

- Overview
- Limitations
- Deploying the Plug-In

Overview

The attribute value uniqueness plug-in verifies and enforces the uniqueness of any number of attributes. To do this, it communicates with attribute value uniqueness plug-ins on other servers. An operation that modifies an attribute being monitored will receive an error if the new value already exists in any of the master replicas. This guarantees that all attributes being verified will have unique values in all the servers, even in the delay between replication updates.

CAUTION The attribute value uniqueness plug-in is classified as an Unstable interface according to the definition provided in `attributes(5)` on Solaris™ systems. It is provided to give you early access to a new technology as an interim solution to multi-master attribute uniqueness. Deployment of this plug-in is not recommended in production environments and will not be supported. Deployment solutions based on this interface may not work with future minor releases of Directory Server. For more information, see “Limitations” on page 380.

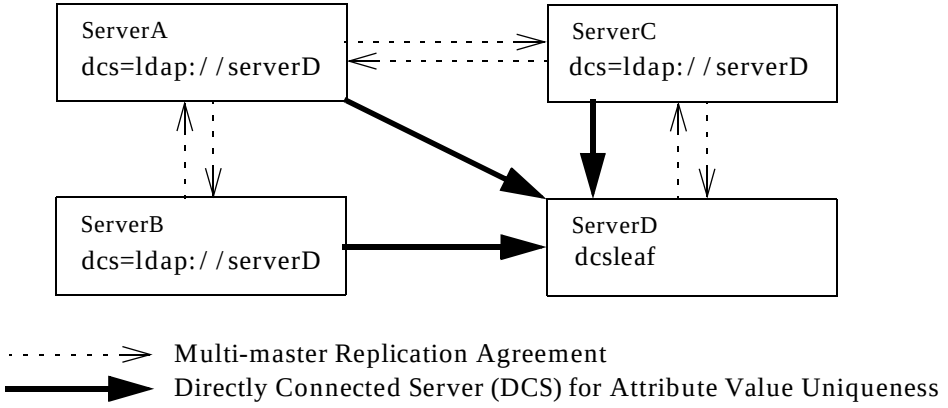
This plug-in is designed to be used only in a multi-master replication environment. Consumer and hub replicas do not accept write operations, and therefore they do not need the attribute uniqueness plug-in to be enabled. Masters in a multi-master configuration contain the same data, but may hold updates that have not yet been replicated. The attribute value uniqueness plug-in must be enabled and configured identically on all replicating master servers as it will contact all masters to verify the latest values for an attribute on any server.

The attribute value uniqueness plug-in verifies uniqueness for any number of attributes. Each attribute may be associated with a list of suffixes where uniqueness will be enforced collectively. The plug-in does not allow “separate” uniqueness within different sets of entries. It ensures that the attribute’s value will be unique in all suffixes in the list and in all master replicas of those suffixes.

Networking Plug-Ins

In order to verify uniqueness on all servers, the attribute value uniqueness plug-in establishes a network of connections, so that all servers can inquire about values on all other servers. The configuration of the plug-in on a server includes a list of Lightweight Directory Access Protocol (LDAP) URLs for nearby servers, called *directly connected servers* or DCS. If one server is configured with the URL of another, the second server will participate without needing the URL of the first. The second server is called a *dcsleaf*. Figure 36-1 illustrates an example of connections between plug-ins.

Figure 36-1 Attribute Value Uniqueness Connections Between Servers



The directly connected servers and leaves are defined in the configuration of the attribute value uniqueness plug-ins. The connections are independent of the replication agreements. The difference between nodes and leaves of a connection diagram is that at startup, nodes initiate the connection to leaves. A node will retry connections for a minute, whereas leaves will listen for 5 minutes for any expected messages.

Connections between the plug-ins are used to determine whether all servers are reachable. By default, the node plug-ins will regularly send *keepalive* messages to the directly connected server every 60 seconds to determine if all master servers are reachable. If a server does not respond to a uniqueness check or a *keepalive* message within a given delay (by default, 30 seconds) it is considered unreachable.

When there are two masters, set the DCS on the primary or more robust master server, and make the other a dcsleaf. This configuration will allow the *keepalive* messages to detect connection problems sooner. With three or four masters, you should avoid having sequences of directly connected servers where one node connects to another node. Instead, make one server a leaf and have all others connect to it, as shown in Figure 36-1. Parameters for configuring communication between plug-ins are described in “Deploying the Plug-In” on page 382.

Distributed Hash Tables

The plug-in architecture optimizes network usage by maintaining tables of attribute values in use. For each attribute being monitored, the network of plug-ins maintains a single table of attribute values in use on all masters. Uniqueness can then be determined by reading a single table instead of performing a search operation on every master.

Attribute values are hashed using the SHA1 hash algorithm. Subsets of the hash space are distributed to the servers in the topology and stored as part of the database containing the suffix. Hashing the values makes the plug-in more secure because copies of attribute values are not sent in clear but avoids the heavier overheads of encryption methods.

Because of the hashing, each plug-in is responsible for a certain subset of the hashed attribute value data. Hashing also ensures that the tables of attribute values are evenly distributed between all master servers. For example, in a topology with 3 masters and 2 attributes configured for uniqueness, the plug-in on each server will manage approximately one third of the values of each attribute. As a result, one third of all uniqueness checks can be performed by the local plug-in and do not require further connections. Another benefit of distribution is that connections between attribute value uniqueness plug-ins are evenly distributed and do not overload any one of the servers.

Limitations

You should also be aware of the performance considerations and potential single points of failure that are inherent when enforcing attribute uniqueness in a multi-master environment. These issues are explained in the following sections.

Using UID Uniqueness Plug-In

The attribute value uniqueness plug-in does not adequately replace the UID uniqueness plug-in provided with Sun™ ONE Directory Server. For example, the attribute value uniqueness plug-in cannot be configured to enforce uniqueness on a single server nor on a single master replica. For more information about the supported plug-in, see Chapter 15, "Using the UID Uniqueness Plug-In," in the *Sun ONE Directory Server Administration Guide.*

CAUTION The attribute value uniqueness plug-in will conflict with instances of the UID uniqueness plug-in. The two plug-ins must not be enabled simultaneously.

No Signature File

No signature file is provided with the attribute value uniqueness plug-in because it is not a supported feature of the DSRK. In order to use this plug-in, you must configure your server to accept unsigned plug-ins.

NOTE For security, you should still verify plug-in signatures and monitor plug-ins that have an invalid signature. For more information, see "Verifying Plug-In Signatures" in Chapter 1 of the *Sun ONE Directory Server Administration Guide.*

Multi-Master Replication Topology

The attribute value uniqueness plug-in is designed only to enforce attribute value uniqueness in multi-master replication scenarios. You must configure your multi-master replication topology before enabling the plug-in. Attribute uniqueness is enforced in the entire replicated suffix, not in arbitrary subtrees of the directory. Subsuffixes must be explicitly specified in the plug-in configuration.

The deployment of attribute value uniqueness also requires a stable topology. You must not add, remove, promote or demote a master server into or out of the multi-master replication topology while the plug-in is enabled. Instead, you must stop all servers, reconfigure your topology, reconfigure the plug-ins and their connections, and then restart all servers.

Unavailability at Startup

When the attribute value uniqueness plug-in is first enabled, the network of plug-ins must create the table of values for each monitored attribute. Hashing all values of all attributes being monitored for uniqueness and distributing those values requires some time at initialization. First, the plug-in must scan all attribute values and build the hash tables. During this scan, all masters will be entirely read-only because the plug-in must ensure uniqueness in the existing attribute values. The time for which the servers will refuse modify operations depends on the number of attributes and the number of entries containing these attributes, but it is typically several minutes.

When the scan of attribute values is complete, modification operations are allowed again. However, server and network activity may be high for a certain period while the tables of hashed values are distributed to all servers. Again, the duration of this activity depends on the number of attribute values being hashed and distributed, but it is typically on the order of several minutes, more for communication over a wide area network (WAN). To reduce startup time to a minimum, we recommend that you enable the plug-in before populating your suffixes. Then populate your suffixes by importing them through LDAP operations.

Performance Considerations

Enforcing attribute value uniqueness in a multi-master topology involves additional communication between servers. Before any update operation can be allowed to proceed, the network of plug-ins must determine if the new value of a given attribute is already used on any of the other masters. The plug-in architecture optimizes this verification procedure, but it may still lead to measurable overhead on update operations.

When using the attribute value uniqueness plug-in, all update operations will require communication between plug-ins over the network. Distributed tables of hashed attribute values optimize this process but every update operation still requires one or more network connections to check for uniqueness and to update the tables. This will both induce delays in the completion of the operation and increase network traffic.

For example, using the attribute value uniqueness plug-in in a multi-master environment over a WAN may significantly increase update operation times. Multi-master replication over WAN uses delayed and grouped operations for efficient network usage. The nature of attribute uniqueness requires immediate network communication between plug-ins, during which time the client operation is blocked. The delay in operations depends on the bandwidth, latency, and network traffic on the WAN.

Single Point of Failure

An unavoidable side-effect of multi-master uniqueness in general is that every server is a single point of failure. If a server is down or unreachable, it is impossible to check the uniqueness of attribute values stored on that server. The hashing mechanism alleviates this problem but does not eliminate it. Without hashing, every uniqueness check would fail if one server is down. With hashing and distribution, a uniqueness check will fail only if the value being checked happens to be stored on the unavailable server. For example, if one of 3 master servers is offline, roughly one third of all uniqueness checks will fail.

The attribute value uniqueness plug-in allows you to configure whether uniqueness is critical or non-critical for a given attribute. When uniqueness of an attribute is critical and a uniqueness check fails because a server is unreachable, the modification operation is refused. Examples of critical attribute values might be user ID, telephone number, credit card number, or e-mail address. The default behavior of the plug-in, is to consider all attributes as critical, and it will not authorize write operations to proceed unless an attribute value can be verified. (When an attribute being checked for uniqueness is configured as non-critical and a uniqueness check fails, the plug-in will report a warning in the error log but allow the modification operation to proceed.)

CAUTION When you have configured attribute values to be non-critical, you should monitor the error logs for these warnings and manually determine uniqueness of attributes that were added or modified while a server was unreachable.

Deploying the Plug-In

Before you deploy the attribute value uniqueness plug-in, you must fully configure your replication topology. All masters must have replication enabled and their replication agreements configured. All replicas should be initialized and active. Then you must create a new instance of the plug-in in every master server. The

configuration of attributes and suffixes where uniqueness is enforced must be identical in every plug-in. This implies that all of the suffixes and subsuffixes listed for all of the attributes must be replicated between every one of the masters. You must create the plug-in configuration entry on all masters before restarting all servers simultaneously.

After you create the plug-in instance, you may use either the Console or the command-line to modify its configuration. If you change the attributes or suffixes where uniqueness is enforced, or if you add or remove a master server from the topology, you must modify all plug-ins and restart all servers simultaneously.

Creating the Plug-In Instance

To create a new plug-in instance, you must create a new entry under `cn=plugins,cn=config` in the directory. The following procedure shows how to do this from the command-line.

1. Use the `ldapmodify` command to add the configuration entry of the new attribute value uniqueness plug-in instance.

NOTE The first part of the command is shown below. The rest of the command is shown in the following steps.

```
ldapmodify -a -h host -p port -D "cn=Directory Manager" -w password
dn: cn=Attribute Value Uniqueness,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Attribute Value Uniqueness
nsslapd-pluginDescription: Unique attribute values across
 multiple servers
nsslapd-pluginType: preoperation
nsslapd-plugin-depends-on-type: database
nsslapd-pluginVersion: 5.2
nsslapd-pluginVendor: Sun Microsystems, Inc.
nsslapd-pluginId: UniqueValue
nsslapd-pluginPath: DSRK_base/lib/valueunique-plugin.extension
nsslapd-pluginInitfunc: valueunique_init
nsslapd-pluginEnabled: on
...
```

Where:

- The *DSRK_base* is the location where you installed the DSRK and the library *extension* depends on your platform.

Alternatively, you may copy the `valueunique-plugin.`*extension* file from the DSRK into *serverRoot*`/lib` and specify that as the plug-in path. If you run your server in 64-bit mode, you must instead copy the *DSRKpath*`/lib/64/valueunique-plugin.`*extension* file into *serverRoot*`/lib/64` where the server will find it automatically. In both 32- and 64-bit modes, the plug-in using the copied library must be configured as follows:

`nsslapd-pluginPath:` *serverRoot*`/lib/valueunique-plugin.`*extension*

For more information, see "64-bit Plug-In Locations" in Chapter 3 of the *Sun ONE Directory Server Plug-In Programming Guide*.

2. The rest of the command specifies the arguments that configure the plug-in. The arguments must be named sequentially starting with 0 (zero).

```
nsslapd-pluginarg0: first_argument
nsslapd-pluginarg1: second_argument
nsslapd-pluginarg2: third_argument
...
nsslapd-pluginargN: last_argument
^D
```

The plug-in arguments are described in the following substeps. Whenever an equals sign (=) appears in an argument, there must be no white space before or after the sign.

a. Define every attribute whose values you wish to be checked for uniqueness. After each attribute, list the base DN of the replicated suffixes where uniqueness of that attribute will be enforced collectively. A given attribute value may occur only once in all listed suffixes.

You must list subsuffixes explicitly, even if their parent suffix is listed. Optionally, specify whether the attribute is critical or not. These plug-in arguments have the following format:

```
nsslapd-pluginargn: attribute=attributeTypeName
[nsslapd-pluginargn+1: onerror=fail or onerror=continue]
nsslapd-pluginargn+2: suffix=suffix1BaseDN
nsslapd-pluginargn+3: suffix=suffix2BaseDN
...
```

When you specify `onerror=fail`, the attribute is considered critical and any inability to determine uniqueness because of an unavailable server will cause the operation to fail. By default, all attributes are considered critical if the `onerror` argument is omitted. Specify `onerror=continue` to make an attribute non-critical. In this case, the inability to determine uniqueness will not cause an operation to fail, but the plug-in will log a warning in the error log.

You must configure the plug-in on all master servers to check uniqueness for all of the same attributes. Also, the `onerror` setting and the suffix DNs of a given attribute configuration must be the same on all servers.

b. After all attributes and their suffixes have been specified, add the following arguments to define the connections between plug-ins:

```
nsslapd-pluginargn: dcsleaf
or
nsslapd-pluginargn: dcs=ldap://host1:port1/
nsslapd-pluginargn+1: dcs=ldap://host2:port2/
...
```

The directly connected server (DCS) definitions should be different on every master so that all masters are connected. For more information, see “Networking Plug-Ins” on page 378. You may specify secure ports if SSL is enabled on the local server and the target host. When using secure ports, all communication between the two plug-ins will be encrypted.

c. Optionally, specify the either or both of the following attributes to define how communications between connected servers are handled. These parameters are specific to each server, depending upon your network conditions:

```
nsslapd-pluginargn: alivecheck=seconds
nsslapd-pluginargn+1: optimeout=seconds
...
```

If the local plug-in has not received a message from a connected server within the delay specified by the `alivecheck` parameter, the plug-in will send a *keepalive* message to the server. After sending either a uniqueness check or a keepalive message to a server, the `optimeout` parameter determines the delay that the plug-in will wait for an answer. If the server does not respond within this delay the server is assumed to be unreachable and the uniqueness check will fail.

The `alivecheck` parameter has no effect if `dcsleaf` is specified, because leaf plug-ins do not send keepalive messages. The default `alivecheck` is 60 seconds, and the default `optimeout` is 30 seconds. When using the plug-in with master servers connected over a WAN, you should set the `alivecheck` and `optimeout` arguments to values that allow for both the average and exceptional delays that occur over your WAN.

3. If you verify plug-in signatures in your server, you must change the configuration to only flag plug-ins with invalid signatures. For details of this procedure, see "Verifying Plug-In Signatures" in Chapter 1 of the *Sun ONE Directory Server Administration Guide*.

4. Before restarting the server, repeat this procedure to configure the attribute value uniqueness plug-in on every master server in your multi-master replication topology.

5. Restart all master servers in your multi-master replication topology. Begin with servers that are a dcsleaf in the plug-in connection diagram, and then restart the ones connected to those leaves.

   The master servers will restart in read-only mode until the plug-ins have initialized and then automatically switch to allow normal write operation. However, network traffic between the plug-ins will still be high while the plug-ins distribute the hashed attribute values. For more information, see "Unavailability at Startup" on page 381.

Configuring the Plug-In Using the Console

After the new instance of the attribute value uniqueness plug-in has been created, you can modify its configuration at any time using the Console. For example, if you add or remove a master server from the replication topology, you must modify the connections between plug-ins to account for this change. You may also modify the attributes or suffixes where uniqueness is enforced by identically reconfiguring all plug-ins in the multi-master topology. (In both of these cases, you must restart all master servers after all modifications have been made.) You may also modify the communication arguments of a plug-in. These are specific to each server and do not require restarting all master servers.

1. On the top-level Configuration tab of the Directory Server console, expand the Plug-Ins node and select the attribute value uniqueness plug-in.

2. In the right-hand panel, select or deselect the checkbox to enable or disable the plug-in.

3. If you have changed the installation path of the DSRK, modify the field for the the plug-in module path accordingly. Do not modify the name of the initialization function.

4. Use the text fields and the Add and Delete buttons to modify the plug-in arguments that configure the plug-in. The syntax for the order of the arguments is the following:

   a. Specify the attributes for which uniqueness is enforced, each one followed by the optional `onerror` argument and one or more suffixes or subsuffixes given by their base DN.

   `attribute=`*attributeTypeName*
   `[onerror=fail` *or* `onerror=continue]`
   `suffix=`*suffixBaseDN*
   `...`

   When `onerror=fail` and uniqueness cannot be checked because a server is unreachable, the modify operation will fail. When `onerror=continue` in the same condition, an error will be logged, the modify operations will succeed, but you must then verify manually on all servers that attribute values are still unique. When this argument is omitted, the default behavior is the same as `onerror=fail`.

   b. Use the following directly connected server (DCS) arguments to specify the connections between attribute value uniqueness plug-ins. Specify that a server is either a leaf or give the LDAP URL of another server. For more information, see “Networking Plug-Ins” on page 378.

   `dcsleaf` *or*
   `dcs=ldap://`*host*`:`*port*`/`
   `...`

   You may specify secure ports if SSL is enabled on the local server and the target host. When using secure ports, all communication between the two plug-ins will be encrypted.

   c. Optionally, specify the either or both of the following attributes to define how communications between connected servers are handled.

   `[alivecheck=`*seconds*`]`
   `[optimeout=`*seconds*`]`

The `alivecheck` parameter determines how often the plug-in will contact its directly connected servers to see if they are still reachable. The default is every 60 seconds. This parameter has no effect if defined on a leaf server. The `optimeout` parameter defines how long the plug-in will wait for a reply to either a uniqueness check or a "keepalive" check before assuming a server is unreachable. The default is 30 seconds.

5. Click Save when you are done editing the plug-in configuration. You will be reminded that you must restart the server for the changes to take effect.
6. If you modified the attribute arguments, including the `onerror` or suffix configuration you must modify the plug-in configuration on all other master servers in exactly the same way. If you modified the directly connected server arguments, modify other plug-in configurations to reestablish a connection between all plug-ins.
7. If you modified the attribute arguments or the directly connected server arguments, you must restart all master servers in the replication topology. If you modified only the connection arguments, you need only to restart this server.

   If you restart all masters, begin with servers that are a `dcsleaf` in the plug-in connection diagram, and then restart the ones connected to those leaves. See also "Unavailability at Startup" on page 381.

Configuring the Plug-In From the Command-Line

The following procedure describes how to enable and configure the attribute value uniqueness plug-in using the `ldapmodify` command. By convention, the DN of the plug-in entry is `cn=Attribute Value Uniqueness,cn=plugins,cn=config`.

1. Enable or disable the plug-in by setting the `nsslapd-pluginEnabled` attribute to `on` or `off`, respectively:

   ```
   ldapmodify -h host -p port -D "cn=Directory Manager" -w password
   dn: cn=Attribute Value Uniqueness,cn=plugins,cn=config
   changetype: modify
   replace: nsslapd-pluginEnabled
   nsslapd-pluginEnabled: on or off
   ^D
   ```

2. Modify the plug-in arguments to set the new configuration of the plug-in. The order of the plug-in arguments is important, therefore you may need to modify many arguments that occur after one that is modified. For example, given the following configuration entry:

```
dn: cn=Attribute Value Uniqueness,cn=plugins,cn=config
...
nsslapd-pluginarg0: attribute=mail
nsslapd-pluginarg1: suffix=dc=example,dc=com
nsslapd-pluginarg2: attribute=uid
nsslapd-pluginarg3: suffix=dc=example,dc=com
nsslapd-pluginarg4: dcs=ldap://host:port/
```

the following command sets the `onerror` parameter for the `mail` attribute and defines the `optimeout` parameter to increase it from the default of 30 seconds:

```
ldapmodify -h host -p port -D "cn=Directory Manager" -w password
dn: cn=Attribute Value Uniqueness,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginarg2
nsslapd-pluginarg2: onerror=continue
-
replace: nsslapd-pluginarg3
nsslapd-pluginarg3: attribute=uid
-
replace: nsslapd-pluginarg4
nsslapd-pluginarg4: suffix=dc=example,dc=com
-
add: nsslapd-pluginarg5
nsslapd-pluginarg5: dcs=ldap://host:port/
-
add: nsslapd-pluginarg6
nsslapd-pluginarg6: optimeout=60
^D
```

See Step 4 of “Configuring the Plug-In Using the Console” on page 386, for a description of the possible argument values and the order of the arguments.

3. If you modified the attribute arguments, including the `onerror` or suffix configuration you must modify the plug-in configuration on all other master servers in exactly the same way. If you modified the directly connected server arguments, modify other plug-in configurations to reestablish a connection between all plug-ins.

4. If you modified the attribute arguments or the directly connected server arguments, you must restart all master servers in the replication topology. If you modified only the connection arguments, you need only to restart this server.

   If you restart all masters, begin with servers that are a `dcsleaf` in the plug-in connection diagram, and then restart the ones connected to those leaves. See also “Unavailability at Startup” on page 381.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X